NECパーソナルコンピュータ
PC-9800シリーズ

NEC

# Software Library

# N88-日本語BASIC(86)(MS-DOS版)6.0 ユーザーズマニュアル

# Software Library

## N88-日本語BASIC(86)
## (MS-DOS版)6.0
## ユーザーズマニュアル

**輸出する際の注意事項**

本製品（ソフトウェアを含む）は，外国為替および外国貿易管理法の規定により，戦略物資等輸出規制品に該当します．従って，日本国外に持出す際には日本国政府の輸出許可申請等必要な手続きをお取り下さい.



BA8B1C

# はじめに

$N_{88}$-日本語BASIC(86)(MS-DOS版)は日本語MS-DOS上で動作するインタプリタシステムで，MS-DOS上の1つのアプリケーションとして位置づけられています．

$N_{88}$-日本語BASIC(86)(MS-DOS版)言語は，PC-9800シリーズパーソナルコンピュータに標準OSとして使用されている$N_{88}$-日本語BASIC(86)と同等あるいはそれ以上の機能，操作性，処理性能を有しており，互換性の高いものです．

$N_{88}$-日本語BASIC(86)で開発されたプログラムをファイルコンバータによってMS-DOSファイルに変換することにより，そのプログラムにわずかな修正を加えるだけで$N_{88}$-日本語BASIC(86)(MS-DOS版)上で実行することが可能です．

また，$N_{88}$-日本語BASIC(86)コンパイラシステムは$N_{88}$-日本語BASIC(86)(MS-DOS版)と機能あるいは処理能力の面で同等のものを持っており，$N_{88}$-日本語BASIC(86)コンパイラでコンパイルされたプログラムは，$N_{88}$-日本語BASIC(86)(MS-DOS版)インタプリタシステム上のプログラムに比べ，2～4倍の速度で実行することができます．

しかも，$N_{88}$-日本語BASIC(86)(MS-DOS版)で作成，編集，デバッグしたプログラムは一般的にそのまま$N_{88}$-日本語BASIC(86)コンパイラのソースプログラムとすることができます．

本書は，$N_{88}$-日本語BASIC(86)(MS-DOS版)および$N_{88}$-日本語BASIC(86)コンパイラの動作環境，操作法あるいは$N_{88}$-日本語BASIC(86)(MS-DOS版)言語機能を使用した具体的なプログラミングの方法などに関して詳細に記述した説明書です．

# このマニュアルの構成

本書は次の2部と付録から構成されています.

## 第1部　$N_{88}$-日本語BASIC(86)(MS-DOS版)インタプリタシステム

$N_{88}$-日本語BASIC(86)(MS-DOS版)インタプリタシステムの起動方法，動作環境，操作方法および$N_{88}$-日本語BASIC(86)(MS-DOS版)言語機能の代表的なものについてその使用方法を説明しています.

ここでBASICとは，$N_{88}$-日本語BASIC(86)(MS-DOS版)および$N_{88}$-日本語BASIC(86)コンパイラの総称を意味します.

なお，ここで断わりなく網の掛けられた部分(▒▒)については，コンパイラに関する記述です.

## 第2部　$N_{88}$-日本語BASIC(86)コンパイラシステム

$N_{88}$-日本語BASIC(86)コンパイラの起動方法，動作環境，操作方法およびコンパイラでコンパイルされたプログラムの起動方法，動作環境について説明しています.

## 付　録

$N_{88}$-日本語BASIC (86) と$N_{88}$-日本語BASIC(86)(MS-DOS版)のノーマルモードとの機能差，ノーマルモードとハイレゾモードとの機能差，およびインタプリタとコンパイラの機能差の一覧表と，その他の資料です.

なお，$N_{88}$-日本語BASIC(86)(MS-DOS版)言語のインタプリタ，コンパイラの共通部分については第1部のインタプリタの解説の項で解説されています. コンパイラシステムを効果的に使用するためにはインタプリタシステムの機能を理解することが必要ですので，第1部，第2部をともにお読みください.

# このマニュアルについての注意点

本書の説明内容はMS-DOSに関してある程度の知識がある方を前提とした内容になっています．たとえば，$N_{88}$-日本語BASIC(86)(MS-DOS版)，$N_{88}$-日本語BASIC(86)コンパイラ，あるいはコンパイルされたプログラムは，いずれもMS-DOS上の1つのアプリケーションプログラムとしてMS-DOS上で動作します．したがって，MS-DOSの基本的な操作方法を理解しておく必要があります．

また，$N_{88}$-日本語BASIC(86)(MS-DOS版)の言語機能の一部にはMS-DOSと関連性の強いものがあります．ディスクファイル関連機能，あるいはチャイルドプロセス制御機能がその代表的なものです．これらの機能の解説中には関連するMS-DOSのコマンドが引用されています．

MS-DOSに関する詳細な内容はMS-DOSのマニュアルを参照してください．

$N_{88}$-日本語BASIC(86)(MS-DOS版)言語の言語機能(個々の命令の機能)の詳細については$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアルを参照してください．

**●2つの画面モード**

$N_{88}$-日本語BASIC(86)(MS-DOS版)言語は，ハードウェアに依存する次の2つのモードを持っています．

- ノーマルモード：PC-9800シリーズの最大640×400の解像度を持つグラフィック画面です．
- ハイレゾモード：XL/XAシリーズの1120×936の解像度を持つグラフィック画面です．なお，ハイレゾモードとはハイレゾリューション(高精細)モードの略称です．

選択できるモードは，ユーザーが使用するコンピュータ，ディスプレイの種類によって異なります．各モードが使用できる機種は次のとおりです．

- ノーマルモード：PC-9800シリーズとPC-98XLシリーズ
- ハイレゾモード：PC-98XL/XAシリーズ

なお，2つのモードを選択できる機種の場合，$N_{88}$-日本語BASIC(86)(MS-DOS版)はコンピュータ本体の動作モード切り替えスイッチの設定によってモードを自動判別します．

本書はノーマルモードとハイレゾモードの固有の解説については次のようなマークをつけています．

Ⓝ：ノーマルモード

Ⓗ：ハイレゾモード

このマークや特に断わりのない場合の記述は，両方のモードに共通の解説であることを示しています．

# このマニュアルについての注意点

# 目　次

## 第1部　$N_{88}$-日本語BASIC(86)(MS-DOS版) インタプリタシステム

## 第11章 拡張機能 143

# 第2部 $N_{88}$-日本語BASIC(86)コンパイラシステム

## 第4章　オブジェクトプログラムの実行　227



## 付　録　231

# 第1部

## $N_{88}$-日本語BASIC(86)(MS-DOS版)

## インタプリタシステム

# 第1章

# インタプリタシステムの概要

$N_{88}$-日本語 BASIC(86)(MS-DOS版)インタプリタシステムの概要を解説します.

## 1.1 インタプリタの提供形態

BASICのディスク(提供媒体)内のファイルとその内容を次の表に示します.

| ファイル名 | 内　容 |
|---|---|
| N88BASIC.EXE | $N_{88}$-日本語 BASIC(86)(MS-DOS版)インタプリタ本体 |
| GPIB.EXE | GP-IB 機能制御部*1 |
| SOUND.EXE | サウンド機能制御部*1 |
| NETWORK.EXE | BRANCH4670 対応ネットワーク機能制御部*1 |
| MSNET.EXE | MS-NETWORKS 機能制御部*1 |
| PHONE.EXE | 電話機能制御部*1 |
| DECCAL.EXE | 10進演算機能制御部*1 |
| LIBMAINT.COM | インタプリタシステムの機能強化の際に使用するユーティリティ. 通常は使用しないため, そのまま保存しておく |
| N88SETUP.EXE | 画面ハードコピー, RS-232Cの諸設定, 演算モードの機能を設定するユーティリティ*2 |
| SETUP.INI<br>ICP.EXE<br>INSTALL.CMF<br>INSTALL.BAT | BASICをMS-DOSのシステムディスクに組み込む際に使用するファイル*3 |

＊1　詳細については「第11章　拡張機能」を参照してください.
＊2　使用方法については,「12.2　セットアップユーティリティ」を参照してください.
＊3　詳細については,「12.1　BASICの登録」を参照してください.

## 1.2 インタプリタシステムの特徴

$N_{88}$-日本語BASIC(86)(MS-DOS版)はMS-DOS上で動作するBASICインタプリタシステムで,PC-9800シリーズパーソナルコンピュータで広く使用されている$N_{88}$-日本語BASIC(86)と機能，操作性で共通性を持ち，処理能力の面で優れています.

▶参照　$N_{88}$-日本語BASIC(86)との相違点→「付録A　$N_{88}$-日本語BASIC(86)と$N_{88}$-日本語BASIC(86)(MS-DOS版)との機能差一覧」

$N_{88}$-日本語BASIC(86)(MS-DOS版)はMS-DOS上で動作することから次のような特徴があります.

(1) MS-DOS上で動作する種々のアプリケーションとデータ交換が可能です.
(2) MS-DOS上で動作するアプリケーションとBASICプログラムの連係動作が可能です.
(3) MS-DOSが提供している種々のファンクション／ユーティリティを使用することが可能です.
(4) BASICで作成・デバッグしたプログラムを$N_{88}$-日本語BASIC(86)コンパイラでコンパイルすることにより，プログラムの高速実行が可能となります.

なお，BASICには，ノーマルモードとハイレゾモードがあります．この2つのモードは，コンピュータの種類によってBASIC起動時に自動的に判別されます.

---

注意：ノーマルモードとハイレゾモードでは，グラフィック画面の解像度が異なるため，グラフィック画面，マウス，外字に関する命令に違いがあります.

---

▶参照　モードの相違点→「付録B　ノーマルモードとハイレゾモードとの機能差一覧」

# 第2章

# インタプリタシステムの起動方法

## 2.1　BASIC起動のための準備

$N_{88}$-日本語 BASIC(86)(MS-DOS版)インタプリタシステムを起動するためには，次のような準備が必要です.

### ● MS-DOSの起動

MS-DOSのアプリケーションであるBASICを起動するには，もちろんMS-DOSが起動されている状態でなければなりません.

▶**参照**　MS-DOSの起動方法→「MS-DOSのマニュアル」

### ● BASICシステムディスクの作成

BASICの提供媒体とMS-DOSのシステムディスクを利用して，ユーザー用のBASICシステムディスクを作成します.

詳しくは「12.1　BASICの登録」を参照してください.

▶**参照**　MS-DOSディスクの作成方法→「MS-DOSのマニュアル」

### ● BASICのセットアップユーティリティの実行

BASICで画面ハードコピーを行う場合やRS-232Cの諸設定を行う場合などは，セットアップユーティリティ〝N88SETUP.EXE〞を使用して設定の変更を行います.

▶**参照**　N88SETUP.EXEの使用方法→「12.2　セットアップユーティリティ」

### ● BASICの環境の準備

BASIC上で同時に4個以上のファイルをオープンする場合,〝CONFIG.SYS〞ファイルが必要となります．すでに〝CONFIG.SYS〞ファイルがディスク上に存在する場合には，その中の記述に変更が必要です.

▶**参照**　ファイル同時オープン数→「2.3　起動パラメータの指定」
　　　　CONFIG.SYSファイル→「MS-DOSのマニュアル」

**●自動実行のための準備**

MS-DOSの起動直後にBASICを自動的に起動する必要がある場合(このような起動方法をオートスタートと呼ぶ)，あらかじめエディタなどで"AUTOEXEC.BAT"という名前のファイルを作成して，この中でBASICの起動を指定しておく必要があります．

▶参照　AUTOEXEC.BAT→「MS-DOSのマニュアル」
オートスタートの指定→2.2の「オートスタート」

## 2.2 BASICの起動

BASICをMS-DOS上から起動するには，次の3つの方法があります．

- ・コマンドラインからの起動
- ・メニューによる起動
- ・オートスタート

次に各起動方法について解説します．

### ■コマンドラインからの起動

MS-DOSのコマンドラインから起動する方法です．

▶参照　コマンドライン→「MS-DOSのマニュアル」

たとえば，Aドライブのカレントディレクトリに"N88BASIC.EXE"があるとします．コマンドラインから次の1行を入力し，[↵]キーを押し，BASICを起動します．コマンド名は"N88BASIC"になります．

A>N88BASIC [↵]

BASIC起動後，次のメッセージが画面に表示されます．

```
NEC N-88 BASIC(86) version 6.0
Copyright (C) 1984,88 by NEC Corporation
xxxxxx Bytes free
Ok
█
```

以降は，MS-DOS ではなく，BASIC の環境になります．このように起動パラメータを指定しなかった場合の起動条件は次のとおりです．

| | |
|---|---|
| ファイル同時オープン数 | 2 |
| ファイルのレコードサイズ | 256 バイト |
| 拡張機能 | 使用せず |
| 起動時に実行される BASIC テキスト | なし |

▶参照　起動パラメータ→「2.3　起動パラメータの指定」

## ■メニューによる起動方法

MS-DOS を固定ディスクから起動したとき，〝MENU〟コマンドのメニューから BASIC を起動する方法です．

MS-DOS(Ver 3.1 以降)では，デフォルトで起動時に〝MENU〟コマンドが起動するように設定されています．

〝MENU〟の機能に「アプリケーションの登録」があり，これで BASIC を登録することによって BASIC が〝MENU〟から起動できるようになります．詳しくは「12.1　BASIC の登録」を参照してください．

## ■オートスタート

MS-DOS の起動直後に BASIC を自動的に起動する方法です．これは，〝AUTOEXEC.BAT〟ファイルに BASIC の起動を指定することにより行います．この場合「2.3　起動パラメータの指定」で説明されている種々の情報を設定することも可能です．

〝INSTALL.BAT〟ファイルを使用することにより，フロッピィディスクに BASIC をインストールして BASIC を自動的に起動するようにできます．詳しくは，「12.1　BASIC の登録」を参照してください．

▶参照　AUTOEXEC.BAT →「MS-DOS のマニュアル」

たとえば，MS-DOS 起動直後に BASIC を自動的に起動し，〝START.BAS〟というプログラムをオートスタートさせるためには，次の情報を〝AUTOEXEC.BAT〟ファイル中に書いておきます．

```
N88BASIC  /T:RUN  "START"
```

つまり，通常の起動時にキーボードから入力する情報を，そのまま〝AUTOEXEC.BAT〟ファイルに書き込んでおけばよいのです．

エディタなどで〝AUTOEXEC.BAT〟という名前のファイルを作成して，この中で前記の

ようにBASICの起動を指定すればよいのですが，〝AUTOEXEC.BAT〞を新たに作成する場合であれば，次のような簡単な方法を用いることができます．

①最初にMS-DOSのコマンドラインに次のようにタイプして，[↵]キーを押します．

```
A>COPY  CON  AUTOEXEC.BAT [↵]
```

このコマンドはキーボードからの入力を〝AUTOEXEC.BAT〞ファイルに書き込むものです．なお，〝AUTOEXEC.BAT〞ファイルは，必ず起動用のMS-DOSシステムディスクのルートディレクトリ中に作ります．

②続いて，ファイルの内容の入力になります．次のようにタイプし，[↵]キーを押します．

```
N88BASIC  /T:RUN  "START" [↵]
```

③他に書き込む内容がない場合，[CTRL]キーを押しながら[Z]キーを押したあと，[↵]キーを押します.「N88BASIC　/T：RUN　"START"」というコマンドが書き込まれた〝AUTOEXEC.BAT〞ファイルが作成されました．

④これで，MS-DOSの起動直後にBASICが起動し，さらに〝START.BAS〞というプログラムを実行することができます．BASICの他のプログラムを実行させる時は，前の例の〝START〞の部分を書き換えます．

---

**注意**：〝AUTOEXEC.BAT〞ファイルを作成し，使用すると，通常の日付と時刻の入力手続きが省略されます．この場合，通常のように日付と時刻の入力を行いたければ〝AUTOEXEC.BAT〞ファイル中でBASICの起動のコマンドの前に，DATEコマンドとTIMEコマンドを入れておくとよいでしょう．

---

## 2.3　起動パラメータの指定

BASIC を起動する際に次の 4 種類の起動パラメータを指定することができます.

・〔/F：〈ファイル同時オープン数〉〕
・〔/S：〈レコードサイズ〉〕
・〔/E：〈拡張機能指定 1〉〔,〈拡張機能指定 2〉〕〔,〈拡張機能指定 3〉〕〔,〈拡張機能指定 4〉〕〔,〈拡張機能指定 5〉〕〔,〈拡張機能指定 6〉〕〕
・〔/T：〈BASIC テキスト〉〕

MS-DOS のコマンド名である〝N88BASIC〞と起動パラメータは，区切り記号も含めて 127 文字以内でなければなりません.

```
A＞N88BASIC  /F：4  /S：512  /T：RUN  "TEST"……………  [↵]
|←――――――――――――127 文字以内――――――――――――→|
```

各起動パラメータは省略することができ，省略時は各々のデフォルト値が採用されます.

/T パラメータ以外は，CR(キャリッジリターンコード)，〝,〞，〝/〞，空白が区切り記号とみなされます.

/T パラメータは，CR のみが区切り記号とみなされます．したがって，/T パラメータは最後に記述する必要があります.

/F, /E, /S の指定順序は任意です.

**●〔/F：〈ファイル同時オープン数〉〕**

BASIC が同時に OPEN できるファイル数を 10 進数で指定します．指定できる範囲は 0〜13 です.

BASIC は〈ファイル同時オープン数〉で指定された入出力用のファイルバッファを確保します.

0 を指定した場合，ユーザーがファイルを OPEN することはできません.

/F パラメータが省略された場合，ファイル同時オープン数は 2 となります.

/F パラメータに 4 以上の値を指定する場合は，〝CONFIG.SYS〞ファイルに必要な FILES パラメータを指定しなければなりません.

次の表に従い，エディタなどを用いて〝CONFIG.SYS〞ファイル中の FILES パラメータに所定の値を設定してください.

| ファイル同時オープン数 | FILESパラメータに指定する値 |
|---|---|
| 3以下 | "CONFIG．SYS" ファイルの必要なし |
| 4 | 9 |
| 5 | 10 |
| 6 | 11 |
| 7 | 11 |
| 8 | 13 |
| 9 | 14 |
| 10 | 15 |
| 11 | 16 |
| 12 | 17 |
| 13(最大値) | 18 |

▶参照　入出力用のファイルバッファ→8.1の「ファイルバッファとファイルの同時オープン数」
FILESパラメータ→「MS-DOSのマニュアル」

コンパイラシステムの環境においては，REM　$FILE命令でファイルの同時オープン数を指定します．

▶参照　REM　$FILE→第2部の「2.2　ファイル使用のための制御命令」

●〔/S：〈レコードサイズ〉〕

ランダムファイルのレコードサイズをバイト単位で指定します．指定できる範囲は1～2048です．

BASICの起動時に，〈レコードサイズ〉で指定されたサイズに相当する入出力ファイルバッファが，/Fパラメータで指定された〈ファイル同時オープン数〉分だけ確保されます．ただし，〈レコードサイズ〉に256より小さな値が指定された場合でも，バッファは256バイト確保されます．

/Sパラメータの指定が省略された場合，レコードサイズは256バイトとなります．

なお，シーケンシャルファイルアクセス時の入出力も〈レコードサイズ〉で指定したサイズ単位で行われます．

---

**注意**：入出力ファイルバッファを使用すると，その分だけ利用者メモリが減りますので，不用意にレコードサイズに大きな値を指定しないようにしてください．

---

コンパイラシステムの環境においては，REM　$FILE命令でファイルのレコードサイズを指定します．

▶参照　REM　$FILE→第2部の「2.2　ファイル使用のための制御命令」

●〔/E：〈拡張機能指定1〉〔，〈拡張機能指定2〉〕〔，〈拡張機能指定3〉〕〔，〈拡張機能指定4〉〕〔，〈拡張機能指定5〉〕〔，〈拡張機能指定6〉〕〕

GPIB，サウンド，電話，BRANCH4670やMS-NETWORKSなどのネットワーク機能，あるいは10進演算機能などの制御部はBASIC本体には含まれていません．これらの制御部は次のような別ファイルに収められています．

▶**参照**　制御部のファイル→「1.1　インタプリタの提供形態」

GPIB.EXE ………GP-IB機能制御部が格納されたファイル
SOUND.EXE ……サウンド機能制御部が格納されたファイル
NETWORK.EXE …BRANCH4670対応ネットワーク機能制御部が格納されたファイル
MSNET.EXE ……MS-NETWORKS機能制御部が格納されたファイル
PHONE.EXE ……電話機能制御部が格納されたファイル
DECCAL.EXE ……10進演算機能制御部が格納されたファイル

/Eパラメータでこれらの拡張機能の使用を宣言すると，これらのファイルから制御部がロードされ，メモリ上で本体と結合されます．

---

**注意**：これらのファイルは起動時に，カレントディレクトリ中に格納されていなければなりません．

---

指定方式は，次のとおりです．〔ドライブ名：〕を省略した場合には，カレントドライブを指定したとみなされます．

〔ドライブ名：〕{ GPIB | SOUND | NETWORK | MSNET | PHONE | DECCAL }

コンパイラシステムの環境においては，拡張機能の制御部がランタイムライブラリに含まれています．このため，インタフェースボードがセットされていれば，拡張機能の使用宣言をしなくても拡張機能を使用することができます．また10進演算機能はそのまま使用できます．ただし，電話制御機能はコンパイラシステムでは使用できませんので注意してください．

●〔/T：〈BASIC テキスト〉〕

BASIC 起動時に自動実行する〈BASIC テキスト〉を指定します.

この指定がある場合,BASIC は起動されると直ちにこのテキスト内に記述された命令を実行します.

/T パラメータは，他の起動パラメータより後に記述しなければなりません.

テキスト内にはマルチステートメント(一行に複数の命令を記述すること)の指定も可能です.

〈起動パラメータの例〉

例 1）A>N88BASIC　/S：512

| | |
|---|---|
| ファイル同時オープン数 | 2 |
| ファイルのレコードサイズ | 512 バイト |
| 拡張機能 | 使用せず |
| 起動時に実行される BASIC テキスト | なし |

例 2）A>N88BASIC　/F：4　/S：64　/T：RUN　"START"

| | |
|---|---|
| ファイル同時オープン数 | 4 |
| ファイルのレコードサイズ | 64 バイト |
| 拡張機能 | 使用せず |
| 起動時に実行される BASIC テキスト | RUN　"START" |

例 3）A>N88BASIC　/E：GPIB，B：MSNET

ここで，GPIB.EXE，MSNET.EXE はルートディレクトリ中になければなりません.

| | |
|---|---|
| ファイル同時オープン数 | 2 |
| ファイルのレコードサイズ | 256 バイト |
| 拡張機能 | GP-IB，MS-NETWORKS 機能 |
| 起動時に実行される BASIC テキスト | なし |

## 2.4 BASIC の終了

SYSTEM 命令を実行すると BASIC は終了し，MS-DOS に制御がもどります.

このとき，オープンされているファイルはすべてクローズされ，すべての変数，プログラムは消去されます．必要なプログラムはあらかじめディスクにセーブしておくようにしてください.

# 第3章

# BASICの動作モード

## 3.1 ダイレクトモード

BASICが起動された直後の動作モードをダイレクトモードといいます．

このモードでBASICの命令を行番号を付けずに入力すると，直ちにその命令が実行されます．

```
PRINT  HEX$(&H34E-&H13C)⏎………命令を入力します．
212……………直ちに実行され，結果が表示されます．
Ok………………BASICのプロンプト"Ok"が表示されます．
█………………次の命令が入力できる状態です．
```

ダイレクトモードで命令を実行するときには，命令の最後に必ず⏎キーを押します．⏎キーを押すことによって初めて，命令がBASICシステムに送られて実行されます．

ダイレクトモードで入力された命令はメモリに格納されませんので，その場限りの計算や記憶する必要のない命令を実行するのに使います．

なお，ダイレクトモードでマルチステートメント(一行に複数の命令を記述すること)を使用してもかまいません．

## 3.2 プログラムモード

行番号を付けて命令を入力すると，その命令はダイレクトモードのように直ちに実行されず，プログラムを構成する1つの行としてメモリに格納されます．

```
100 '--- PRINT TEST ---⏎
110 CONSOLE ,,,1⏎
120 FOR I=1 TO 7⏎
130  COLOR I⏎
140  FOR J=1 TO 24⏎
150   PRINT "COLOR=>";I;⏎
160 NEXT J,I⏎
170 COLOR 7:END⏎
```

メモリに格納されたプログラムはプログラム実行用の命令(RUN 命令)により実行されます．このように，プログラムを行番号つきで入力し，実行するモードをプログラムモードと呼びます．

プログラムモードでは，BASIC はメモリに格納された命令を順次(行番号の小さい方から大きい方へ)実行していきます．

プログラム実行中にエラーが発生したり，プログラムの実行が終了すると，再びダイレクトモードにもどります．

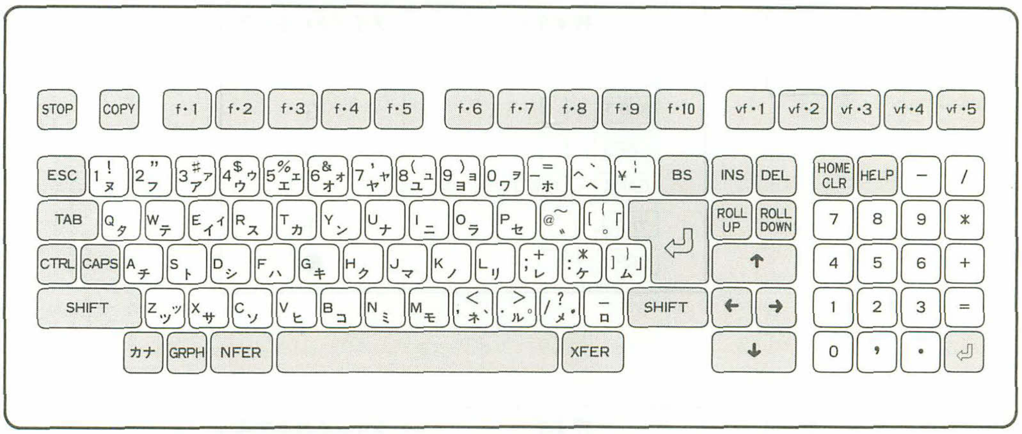
**注意**：コンパイラシステム環境においては，ダイレクトモードという動作モードは存在しません．プログラム実行中にエラーが発生したり，プログラムが終了したりすると，MS-DOS に制御がもどります．

# 第4章

# キーボードとスクリーンエディタ

## 4.1　キーの配置

PC-9800 シリーズのキーボードは JIS 配列に準拠したものです．




キーボードには英文字，カタカナ，数字，記号などの文字をタイプするための文字キーと，その他特殊な役割を持つ特殊キーがあります．図ので示したキーが特殊キーです．

---

**注意**：キーの種類および配置は，機種によって異なります．

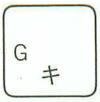
---

## 4.2 文字キー

1つの文字キーで数種類の文字をタイプすることができます．タイプする文字の種類は特殊キーとの組合せで選択することができます．

3つのキーを例にして文字をタイプする方法を説明します．〝+〞と〝,〞は次のことを意味します．

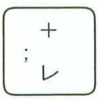
| | |
|---|---|
| [A]+[B] | [A]キーを押しながら[B]キーを押す． |
| [C], | [C]キーを押してロックさせておく． |




| 押すキー | タイプされる文字 |
|---|---|
| □ | g |
| [SHIFT]+□ | G |
| [CAPS], □ | G |
| [CAPS], [SHIFT]+□ | g |
| [カナ], □ | キ |
| [GRPH]+□ | ● |




| 押すキー | タイプされる文字 |
|---|---|
| □ | 4 |
| [SHIFT]+□ | $ |
| [カナ], □ | ウ |
| [カナ], [SHIFT]+□ | ゥ |




| 押すキー | タイプされる文字 |
|---|---|
| □ | ; |
| [SHIFT]+□ | + |
| [カナ], □ | レ |
| [GRPH]+□ | ▮ |

# 4.3 特殊キー

## ■文字とグラフィックシンボルのタイプに関係するキー

**SHIFT （シフトキー）**

このキーを押しながら文字キーを押すと，英大文字(ただし，CAPSキーがロックされているときは英小文字)，あるいは各文字キーの上部に刻印されている文字をタイプすることができます．このキーはキーボードの左右両側にありますがどちらを使ってもかまいません．

**CAPS （キャピタルロックキー）**

英大文字をタイプするのに使います．一度押すとこのキーはロックされた状態になり，英大文字をタイプできるようになります．もう一度押すとロックは解除されます．ロックが解除された状態では英小文字をタイプできます．ただし，SHIFTキーを使用した場合には，ここで説明した英大文字と英小文字の関係は逆になります．

**カナ （カナキー）**

カナ文字をタイプするのに使います．CAPSと同様に一度押すとこのキーはロックされた状態になります．ロックされるとカタカナをタイプできるようになります．もう一度押すとロックは解除され英文字がタイプできる状態にもどります．SHIFTキーを押した状態では ッ，ァ，ィ，ゥ，ェ，ォ(カナの小文字)など各キーの右側に刻印された文字をタイプできます．

---

**注意**：ロックの可能なキーのロック状態は，機種により次の2種類があります．

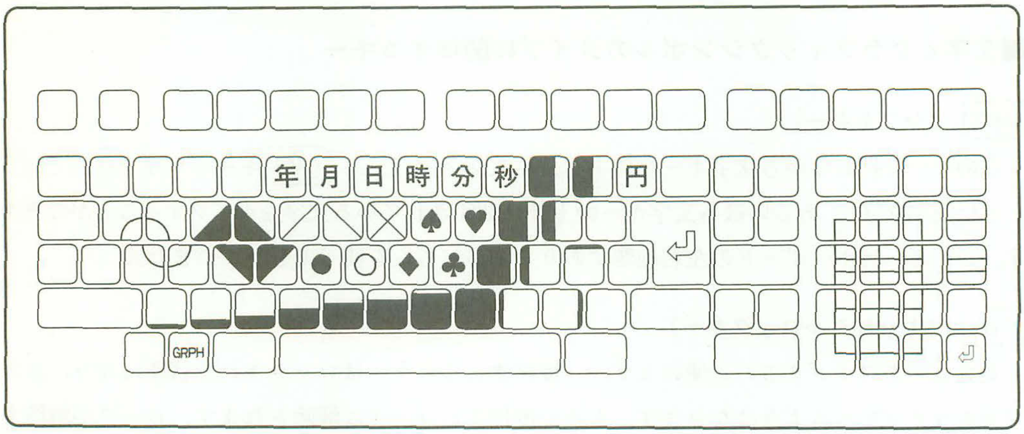
- ロック状態で，そのキーが沈み込むもの
- ロック状態で，キーボード上のLEDが点灯するもの

---

**GRPH （グラフキー）**

このキーを押しながら文字キーを押すとグラフィックシンボルをタイプすることができます．グラフィックシンボルを使う場合，グラフィックキャラクタモードに設定します．

グラフィックシンボルのキー配列




▶参照　グラフィックキャラクタモード→「7.1　日本語キャラクタモードとグラフィックキャラクタモード」

## ■文字編集に関係するキー

[　　]　(スペースキー)

空白をタイプするときに使います.

[⏎]　(リターンキー)

ダイレクトモードではこのキーが押されると命令が実行されます. プログラムモードではこのキーを押すことによって1行の入力が終わります. カーソルは次の行の先頭に移動します.

[TAB]　(タブキー)

行の頭をそろえるときに使います. このキーを押すとカーソルを8桁単位で右へ移動させることができます. なお, このキーでタブコード(&H09)を入力することはできません.

[DEL]　(デリートキー)

カーソルの左の1文字を削除するときに使います. カーソルより右にある文字は左に詰まります.

[BS]　(バックスペースキー)

[DEL]キーと同じ動作をします.

**INS (インサートキー)**

スクリーンエディタは通常重ね書きモードですが，このキーを押すと挿入モードになります．挿入モードで文字をタイプするとカーソルの位置に文字が挿入されていきます．カーソルより右にある文字は順に右に送られます．再び INS キーを押したり，カーソル移動キーや ⏎ キーを押したときに，重ね書きモードにもどります．

**→，←，↑，↓ (カーソル移動キー)**

カーソルを上下左右に動かすのに使います．

**HOME CLR (ホーム・クリアキー)**

このキーを押すとテキスト画面のスクロール画面が消されカーソルはホームポジション(スクロール画面の左上隅)に移ります．SHIFT キーを押しながらこのキーを押すと，画面はそのままで，カーソルだけがホームポジションに移動します．

**CLR (クリアキー)(XL/XA シリーズのみ)**

このキーを押すとテキスト画面のスクロール画面が消されカーソルはホームポジションにもどります．

**HOME (ホームキー)(XL/XA シリーズのみ)**

このキーを押すと，画面はそのままでカーソルだけがホームポジションに移動します．

▶**参照**　スクロール画面→5.1の「スクロール画面の範囲」

## ■その他のキー

**STOP (ストップキー)**

ダイレクトモードの命令やプログラムの実行を強制的に中止します．

**COPY (コピーキー)**

このキーを押すとプリンタで画面のハードコピーをとることができます．

▶**参照**　ハードコピー→「5.3　画面ハードコピー」

**HELP (ヘルプキー)**

プログラムの実行中にエラーが起きた場合，その命令と位置を表示します．

▶**参照**　エラー位置の検出→4.4の「便利な編集コマンドとキー」

**ESC (エスケープキー)**

エスケープコード(&H1B)を入力することができます．

[ROLL DOWN]（ロールダウンキー）

[ROLL UP]（ロールアップキー）

EDIT 命令実行後，プログラム全体を上下にスクロールさせることができます．

これはスクリーンエディタでプログラムの修正を行うのに有効です．また，[ROLL UP]キーを押すと，テキスト画面に対してテキストが上に，[ROLL DOWN]キーを押すと下に移動します．

▶**参照**　EDIT 命令→ 4.4 の「便利な編集コマンドとキー」，「$N_{88}$-日本語 BASIC(86)(MS-DOS版)リファレンスマニュアル」

[XFER]（変換キー）

日本語入力の制御キーとして使います．

▶**参照**　日本語入力 →「日本語入力ガイド」

[NFER]（無変換キー）

日本語入力の制御キーとして使います．

---

**注意**：[NFER]キーは機種によってはありません．

---

[f･1]～[f･10]，[vf･1]～[vf･5]（ファンクションキー）

15 種類のファンクションキーが使用できます．

BASIC の起動時には各ファンクションキーに次に示す文字列が割り当てられています．$^{C}{}_{R}$は，[⏎]キーを押すのと同じ働きがあります．

| | | | | | |
|---|---|---|---|---|---|
| [f･1] | load " | [f･6] | save " | [vf･1] | files |
| [f･2] | auto | [f･7] | key | [vf･2] | renum |
| [f･3] | go to | [f･8] | print | [vf･3] | cls 2 $^{C}{}_{R}$ |
| [f･4] | list | [f･9] | edit . $^{C}{}_{R}$ | [vf･4] | input |
| [f･5] | run $^{C}{}_{R}$ | [f･10] | cont $^{C}{}_{R}$ | [vf･5] | key list $^{C}{}_{R}$ |

これらのファンクションキーを使えば文字列のタイプを短い時間で行うことができます．さらにこのファンクションキーには，利用者が自由に文字列を定義することができます．

それには，BASIC の KEY 命令を使います．

たとえば，

```
KEY 3, "files"+CHR$(13)[⏎]
```

（[⏎]はリターンキーを押すことを表す）

とすると[f･3]キーに，〝files $^{C}{}_{R}$〟が定義されます．

▶**参照**　KEY 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

---

**注意：** 機種によっては，[vf･1]～[vf･5]キーはありません．[vf･1]～[vf･5]キーのある種類において，KEY命令で指定する〈キー番号〉の11～15は，[vf･1]～[vf･5]キーに対応します．
ファンクションキーに定義したグラフィックキャラクタを，日本語キャラクタモードで正しく入力することはできません．グラフィックキャラクタモードでご使用ください．
また，ファンクションキーに2バイト系日本語文字列を定義することはできません．

---

定義されているファンクションキーの内容は，画面の最下行に表示されるようになっています．表示内容は，40桁モードのときと80桁モードのときで異なります．表示される文字数は各キーごとに6文字です．

40桁モード([vf･1]～[vf･5]キーの内容は表示されない)
　[SHIFT]キーを押さない場合　…　[f･1]～[f･5]キーが表示される
　[SHIFT]キーを押した場合　　…　[f･6]～[f･10]キーが表示される
80桁モード
　[SHIFT]キーを押さない場合　…　[f･1]～[f･10]キーが表示される
　[SHIFT]キーを押した場合　　…　[vf･1]～[vf･5]キーが表示される

定義されているファンクションキーの文字列をすべてテキスト画面に表示させるには，KEY LIST命令を実行します．

KEY　LIST [↵]

**[CTRL]　(コントロールキー)**

他のキーと組み合わせて，特殊な機能を実行することができます．

▶**参照**　[CTRL]キーの機能→4.4の「[CTRL]キーを使った便利な編集機能」
日本語キャラクタモード，グラフィックキャラクタモード→「7.1 日本語キャラクタモードとグラフィックキャラクタモード」

---

**参考：**[STOP]，[HELP]，[f･1]～[f･10]キーは，それぞれON STOP GOSUB，ON HELP GOSUB，ON KEY GOSUB命令を使って，キー割り込みの処理を行うことができます．

---

▶**参照**　ON STOP GOSUB命令，ON HELP GOSUB命令，ON KEY GOSUB命令→「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」

### ■テンキーについて

キーボードの右には演算子などのキーとともに数字のキーが並んでいます．これをテンキーといいます．数字を続けて入力する場合などに便利です．

---

**注意**：キーの配置は，機種によって異なります．

---

## 4.4 スクリーンエディタ

BASICにはプログラムの作成および修正が簡単に行えるように，画面編集機能が用意されています．これをスクリーンエディタといいます．

ここでは文字キー，特殊キーを使ってBASICプログラムを編集する方法の概略を説明します．

### ■ BASICプログラムの入力

BASICが起動するとスクリーンエディタの機能が使えるようになり，文字キーを押すと対応する文字が画面のカーソルの位置に表示されます．

BASICのプログラムは必ず命令の前に1から65529の番号をつけて入力しなければなりません．これを行番号といいます．次のように入力してみてください．これを元にしてスクリーンエディタの使い方を説明します．

```
10  PRINT  "Hardware" [↵]
20  PRINT  "Software" [↵]
```

### ■プログラムの修正

#### ●文字の削除，変更，挿入

**文字の削除**

10行目のHardwareを削除するには，まずカーソル移動キーを使って行末の"の位置にカーソルを移動します．この位置で[DEL]キーを押すとeが削除されます．続けて[DEL]キーを7度押すとHardwareがすべて削除されます．削除したら[↵]キーを押します．

**文字の変更**

20行目のSoftwareをComputerに変更するには，カーソル移動キーを使ってカーソルをSの位置まで移動しComputerとタイプします．タイプしたら[↵]キーを押します．

**文字の挿入**

10行目の""の間にSuperと入れるには，まずカーソル移動キーを使ってカーソルを行末の"の位置まで移動し[INS]キーを押します．[INS]キーを押すと挿入モードになります．ここでSuper

とタイプします．挿入モードでは，タイプされた文字はカーソルの前に挿入されていきます．挿入モードを抜けるにはもう一度[INS]キーを押すかカーソル移動キーまたは[⏎]キーを押します．ここでは，[INS]キーを押してください．タイプしたら[⏎]キーを押します．

修正の結果は次のようになります．

```
10 PRINT "Super"
20 PRINT "Computer"
```

---

**注意**：1つの行のプログラムのタイプ•修正が終わったら，必ず[⏎]キーを押してください．[⏎]キーを押すことによって初めて，タイプ•修正した内容がコンピュータのメモリに格納されます．これを入力するといいます．また，[⏎]キーを押すときにはカーソルがその行にあればどの位置にあってもかまいません．カーソルをいちいち行末に移動する必要はありません．

---

**オートリピート機能について**

同じキーを約0.5秒押し続けると同じ文字が連続して表示されていきます．これをオートリピート機能といいます．この機能は文字キーばかりでなく特殊キー(ファンクションキーを除く)にもあります．同じ文字を連続してタイプしたいときに便利です．

### ●行の挿入，削除

**行の挿入**

行を増やしたいときには，現在ある行番号の間の番号，あるいは前後の行番号をつけた命令を入力します．

たとえば，前の例で10行目と20行目の間に他の命令を挿入したい場合には，

```
15 PRINT "Personal" [⏎]
```

と入力します．

**行の削除**

行を削除したいときには，削除したい行の行番号だけを入力します．

たとえば，10行を削除したいときには，

```
10 [⏎]
```

と入力します．

### ●修正結果の確認

修正結果を確認するにはLIST命令を使います．LISTはプログラムを画面表示させる命令です．今までの修正結果を表示してみます．

```
list ⏎
15  PRINT  "Personal"
20  PRINT  "Computer"
Ok
█
```

正しく修正している場合にはこのようになります．このようにならない場合には，今まで使ったキー操作法を使って修正してください．BASICのスクリーンエディタ機能は画面表示されている文字に対して有効ですから，LIST命令で画面表示させた文字の上にカーソルを移動して修正することができます．

▶参照　LIST命令→「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」

## ■便利な編集コマンドとキー

HELPキー

プログラムを実行したときプログラム中にエラーがあると，エラーのある命令のところで実行が中断されます．このときエラーの種類とそのエラーの起きた行番号を示すエラーメッセージが表示されます．ここで，HELPキーを押すとその行が表示され，エラーのある命令かその次のワードの先頭でカーソルが点滅します．なお，ここでワードとは，空白，コンマ，セミコロン，コロンのいずれかで区切られた文字の並びのことです．

```
20  PRINT  A : PLINT  B : GOTO  10
```

この場合エラーは〝PLINT〞の〝L〞で，カーソルはBにあります．カーソル移動キーをLに移動しRをタイプし直して⏎キーを押します．

このとき，LIST　.を使うと，その行だけを表示させることができます．

---

**注意**：エラーが起こってHELPキーを押した場合，必ずしもエラーの発生したワード(行)にカーソルがくるとはかぎりません．

---

### EDIT 命令

プログラムを全体に渡って修正したい場合があります．そのようなときにはEDIT命令を使います．

EDIT　<エラーが起きた行番号>

指定した行が画面の最上位に表示されエディットモードに入ります．このモードでは[ROLL UP]，[ROLL DOWN]キーを使うことにより，プログラム全体をスクロールしながら表示させ，訂正することができます．この場合も各行の訂正が終わったら必ず[⏎]キーを押さなければなりません．

エディットモードは，RUN，GOTOなどでプログラムを実行すると解除されます．

### LIST 命令

プログラムを画面表示します．画面表示は[CTRL]＋[S]で一時停止でき，もう一度[CTRL]＋[S]キーなどを押すと表示が再開されます．[CTRL]＋[C]を実行すると画面表示は中止され，コマンドモードにもどります．

▶参照　[CTRL]＋[S]キー →「[CTRL]キーを使った便利な編集機能」

### RENUM 命令

プログラムの行番号をつけかえます．行番号の間隔や行番号を変更することができます．なお，この命令でプログラムの各行の順番が変わることはありません．

### NEW 命令

メモリ上のプログラムを消去します．新規にプログラムを編集するときやメモリをクリアしておく場合に使います．

## ■CTRLキーを使った便利な編集機能

プログラムを編集するときに文字キーとCTRLキーを組み合わせて使うことによって編集を効率よく行うことができます.

| | |
|---|---|
| CTRL＋A | HELPキーと同じ機能を果たします. |
| CTRL＋B | カーソルを1つ左のワードの先頭に位置づけます. |
| CTRL＋C | STOPキーと同じ. プログラムの実行を強制的に中止します. |
| CTRL＋D | カーソルのある位置からそのワードの終わりまでを削除します. |
| CTRL＋E | カーソルのある位置からその行の終わりまでを消去します. |
| CTRL＋F | カーソルを1つ右のワードの先頭に位置づけます. |
| CTRL＋G | スピーカ(ビープ音)を鳴らします. |
| CTRL＋H | BSキーと同じ. カーソルの左の1文字を削除します. |
| CTRL＋I | TABキーと同じ. タブ位置までカーソルを移動させます. |
| CTRL＋J | ラインフィードを挿入します. 挿入モードのときはカーソルのある位置からその行の終わりまでを分割して次の行に移動します. |
| CTRL＋K | SHIFT＋HOME CLRキーと同じ. カーソルをホームポジションに移動します. |
| CTRL＋L | HOME CLRキーと同じ. テキスト画面をクリアします. |
| CTRL＋M | ↵キーと同じです. |
| CTRL＋O | テキスト画面の表示を中止します. 再びCTRL＋Oを入力すると再開されます. |
| CTRL＋R | INSキーと同じ. 挿入モードになります. |
| CTRL＋S | プログラムの実行を一時停止します. STOP, COPY, SHIFT, カナ, GRPH, CAPS, CTRL, CTRL＋C以外のキーを入力すれば実行が再開されます. |
| CTRL＋U | カーソルのある行を消去します. |
| CTRL＋X | カーソルのある行の行末にカーソルを移動させます. |

---

**注意**：ワードとは, 空白, コンマ(,), コロン(:)のいずれかで区切られた文字の並びのことです.

MS-DOSにおいて, CTRLキーを押しながらファンクションキーを押した場合, 画面クリアや表示行数の変更などの機能が動作します (MS-DOSのマニュアルを参照).

MS-DOS 2.0上でBASICを起動した場合, CTRLキー＋ファンクションキーを押すことによってBASIC起動後も上記の機能が動作します.

BASICの状態に関係なく, CTRL＋ファンクションキーによって画面クリアや表示行数の変更が行われると, BASICのWIDTH命令やCONSOLE命令などで設定した状態と異なる画面表示となりますので, ご注意ください.

---

# 第5章
# 画　面

BASICの画面には大きく分けてテキスト画面とグラフィック画面の2種類があります．テキスト画面には文字を表示させることができます．いっぽう，グラフィック画面には点や線や絵などを表示させることができます．

テキスト画面とグラフィック画面は独立したものです．ディスプレイにどちらか一方だけを表示したり，同時に重ねて表示することが自由にできます．

**●ノーマルモードとハイレゾモード**

BASICにはハードウェアに依存した2つのモードがあります．

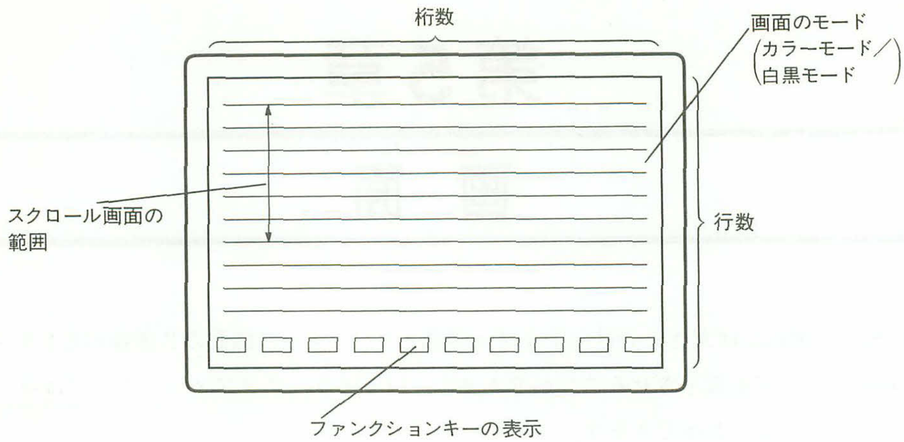
ノーマルモードとハイレゾモードでは画面の解像度に大きな違いがあるため，特にグラフィック画面を制御する命令・関数の使い方が異なります．この章では，各々のモードの場合を分けて記述していますのでご注意ください．なお，特に断り書きがない場合は，両モードに共通な内容を記述しています．

## 5.1 テキスト画面

テキスト画面は文字(キャラクタ)を表示するための画面で，1バイト文字で最大80桁×25行表示することができます．また，文字の色は8色の中から選択することができます．この画面を利用してBASICのプログラムを作成したり，漢字を含む文字を表示することができます．

テキスト画面は次のような項目を設定することができます．

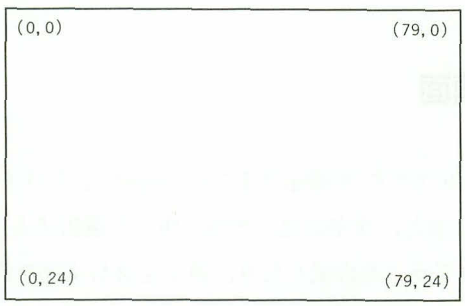
- 桁数と行数
- スクロール画面の範囲
- ファンクションキーの表示
- カラーモード／白黒モード
- テキスト画面の文字の色と表示モード




## ■桁数と行数

キャラクタを画面に出力するためには，行(row)および桁(column)の概念を用います．1つの桁には1バイト文字1文字が入ります．テキスト画面の原点は左上隅の桁で，(0, 0)で表されます．LOCATE命令やCOLOR@命令で参照されるテキスト画面の座標は(桁，行)，つまり(X, Y)の順で表されます．テキスト画面の座標をキャラクタ座標と呼びます．

テキスト画面の座標の例(80桁×25行)




### ●ノーマルモードの場合Ⓝ

テキスト画面には桁数，行数の異なる4つのモードがあります．各モードの桁数，行数はそれぞれ次の表のとおりです．

| モードの名称 | 表示文字数(横×縦) |
|---|---|
| 40桁20行モード | 40×20 |
| 80桁20行モード | 80×20 |
| 40桁25行モード | 40×25 |
| 80桁25行モード | 80×25 |

1つの桁には1バイト文字が1つ入ります．2バイトの日本語文字の場合は表示可能な文字数はこの半分になります．モードを変更するには本体のディップスイッチ，またはWIDTH命令を使います．

ディップスイッチでモードを設定するには本体の電源をOFFにした状態で行います．

| モード | ディップスイッチSW2による設定 | |
|---|---|---|
| | 3番 | 4番 |
| 40桁20行モード | OFF | OFF |
| 80桁20行モード | ON | OFF |
| 40桁25行モード | OFF | ON |
| 80桁25行モード | ON | ON |

ディップスイッチによる指定はBASICが起動した初期状態を設定するもので，WIDTH命令が実行されるまで有効です．

▶**参照**　ディップスイッチ→本体添付の「ガイドブック」

WIDTH命令でモードを設定する場合は次のようにします．

**WIDTH命令**

　　WIDTH　〈桁数〉〔，〈行数〉〕

例）40桁×20行にする．

　　WIDTH　40，20

WIDTH命令による指定はリセットするか，次のWIDTH命令が実行されるまで有効です．

**●ハイレゾモードの場合 Ⓗ**

ハイレゾモードでは80桁×25行のモードのみです．したがってディップスイッチやWIDTH命令の画面に対する指定は無効となります．

## ■スクロール画面の範囲

テキスト画面にデータを表示していくとき，表示データが画面いっぱいになるまでは画面の最上行から画面の下へ向かって1行ずつ順々に表示されていきます．表示されるデータが画面いっぱいになると，最上行にある一番古いデータが消え，残りの画面データは1行上へ移動し，空白になった最下行には最新の表示データが入ります．これが続けて行われると，データが次々に上方へ移動して行くようにみえます．これを画面の〝スクロール〟といいます．

BASICが起動した初期状態では画面全体がスクロールしますが，スクロールする範囲を限定し，スクロールしない部分には同じ内容を常に表示させることができます．この限定されたスクロール範囲のことをスクロール画面と呼びます．

スクロール画面の範囲の指定はCONSOLE命令で行います．CONSOLE命令の1番目のパラメータにスクロール開始行を，2番目のパラメータにスクロール行数を指定します．

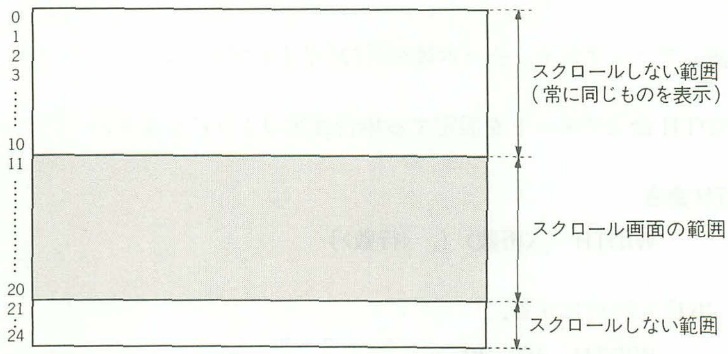
**CONSOLE命令**

**CONSOLE　〔〈スクロール開始行〉〕〔,〈スクロール行数〉〕〔,〈ファンクションキー表示スイッチ〉〕〔,〈カラー／白黒スイッチ〉〕〔,〈キャラクタモード〉〕**

スクロール開始行の指定は画面の最上行から0，1，2，3，4……と数えます．20行のモードの場合は19まで，25行モードの場合は24までの数値を指定できます．

**例)**　25行のモードで最上行から10行目までをスクロールさせないで11行目から20行目までをスクロールさせる．

```
CONSOLE 11, 10
```




## ■ファンクションキーの表示

BASICが起動した初期状態では画面の最下行にはファンクションキーに登録された文字列の一部が表示されていますが，CONSOLE命令を使って表示させないようにすることができます．ファンクションキーを表示する場合にはCONSOLE命令の3番目のパラメータに1を指定し，表示しない場合には0を指定します．

| | |
|---|---|
| CONSOLE ,,0 | 表示しない |
| CONSOLE ,,1 | 表示する |

## ■カラーモード／白黒モード

テキスト画面に表示する文字の色をカラーにするか，白黒にするかを設定することができます．BASIC起動直後は白黒モードになっています．

カラーモード／白黒モードの設定はCONSOLE命令で行います．4番目のパラメータに1を指定するとカラーモードに，0を指定すると白黒モードになります．

```
CONSOLE  , , , 1          カラーモード
CONSOLE  , , , 0          白黒モード
```

ここでいうカラーモード，白黒モードはカラーディスプレイを使うかモノクロディスプレイを使うかということとは別の問題です．たとえば，カラーモードでモノクロディスプレイを使った場合は色は濃淡で表されます．カラーモード，白黒モードとディスプレイの関係は次のようになります．

| テキスト画面のモード | カラーディスプレイ | モノクロディスプレイ |
|---|---|---|
| カラーモード | ○ | △ |
| 白黒モード | ○ | ○ |

○：正常に表示される　△：濃淡で表示される

▶**参照**　キャラクタモード→「7.1　日本語キャラクタモードとグラフィックキャクタモード」

## ■テキスト画面の文字の色と表示モード

テキスト画面に表示する文字の色や表示モードを〔1〕COLOR 命令を使って変えることができます．

〔1〕COLOR 命令

COLOR　〔〈ファンクションコード〉〕〔，〈バックグラウンドカラー〉〕〔，〈ボーダーカラー〉〕〔，〈フォアグラウンドカラー〉〕〔，〈パレットモード〉〕

このなかで，テキストの文字の色を変えるパラメータは１番目の〈ファンクションコード〉です．他のパラメータはテキスト画面には関係なく，グラフィック画面の色の指定に関するものです．

▶**参照**　グラフィック画面の色の指定→「6.3　グラフィック画面の色」

〈ファンクションコード〉の値の意味はテキスト画面がカラーモードになっているか，白黒モードになっているかによって異なります．カラーモードの場合は文字の色を８色の中から選択できます．白黒モードの場合は文字の表示モードを選択できます．

| ファンクションコード | カラーモードの場合（色） | 白黒モードの場合（表示モード） |
|---|---|---|
| 0 | 黒 | ノーマル |
| 1 | 青 | シークレット |
| 2 | 赤 | ブリンク |
| 3 | 紫 | シークレット(1と同じ) |
| 4 | 緑 | リバース |
| 5 | 水色 | リバースシークレット |
| 6 | 黄 | リバースブリンク |
| 7 | 白 | リバースシークレット(5と同じ) |

ノーマルとは通常の表示モードです．リバースにすると背景と文字の色が逆転します．シークレットは入力された文字を表示せず，文字数分だけカーソルが移動していきます．ブリンクは文字を点滅させます．

BASICの起動直後はノーマルになっています．

例1）　テキスト画面の文字を紫にする．

```
CONSOLE  ,,,1……… カラーモードに設定
COLOR  3      ……… 文字の色を紫にする
```

例2）　文字を反転して表示する．

```
CONSOLE  ,,,0……… 白黒モードに設定
COLOR  4      ……… 反転の表示
```

●範囲を指定する場合

〔1〕COLOR命令では，テキスト画面に表示する文字の色あるいは表示のモードの設定はこれから表示させる文字を対象としています．それに対してCOLOR@命令を使うと，すでに表示されている画面上の任意の範囲の文字に対して設定することができます．

COLOR@命令

```
COLOR@  (X1, Y1)-(X2, Y2)〔,〈ファンクションコード〉〕
```

ファンクションコードについては〔1〕COLOR命令の場合と同様です．また，座標の指定はキャラクタ座標で行います．

例）　80桁×25行モードで画面の右上の部分の文字を赤にする．

```
CONSOLE  ,,,1
COLOR@  (40, 0)-(79, 11), 2
```

# 5.2 グラフィック画面

グラフィック画面に絵や画像などを描く際には，場合に応じて最も適した画面を設定しておかなければなりません．BASICのグラフィック画面には，分解能と色の種類(カラー／白黒)の違う複数のモードがあり，その中から適当なものを選択して使用することができます．

## ■画面モード

グラフィック画面には複数の画面モードがあります．

画面モードを変えることによって，カラーで細かいドットによる美しい画像を描いたり，白黒で粗いドットの画像を描いたりすることができます．

### ●画面モードの種類

ノーマルモードの場合 Ⓝ

| 画面モード | 画面のタイプ | 分解能(横×縦) |
|---|---|---|
| カラーモード | カラー | 640×200 |
| 白黒モード | 白黒 | 640×200 |
| 高分解能白黒モード | 白黒 | 640×400 |
| 高分解能カラーモード | カラー | 640×400 |

なお，カラーモードおよび白黒モードの2つを合わせて低分解能モード，他の2つを合わせて高分解能モードと呼ぶこともあります．

ハイレゾモードの場合 Ⓗ

| 画面モード | 画面のタイプ | 分解能(横×縦) |
|---|---|---|
| カラーモード | カラー | 1120×936 |
| 白黒モード | 白黒 | 1120×936 |

### ●画面モードの選択

画面モードの選択はSCREEN命令の1番目のパラメータで行います．

SCREEN命令

SCREEN 〔<画面モード>〕〔, <画面スイッチ>〕〔, <アクティブページ>〕〔, <ディスプレイページ>〕

<画面モード>の指定値とモードの関係は次のとおりです．BASIC起動直後はカラーモードになっています．

ノーマルモードの場合Ⓝ

| 〈画面モード〉の指定値 | 画面モード |
|---|---|
| 0 | カラーモード |
| 1 | 白黒モード |
| 2 | 高分解能白黒モード |
| 3 | 高分解能カラーモード |

例）　高分解能カラーモードを使う場合

```
SCREEN 3
```

ハイレゾモードの場合Ⓗ

| 〈画面モード〉の指定値 | 画面モード |
|---|---|
| 0 | カラーモード |
| 1 | 白黒モード |

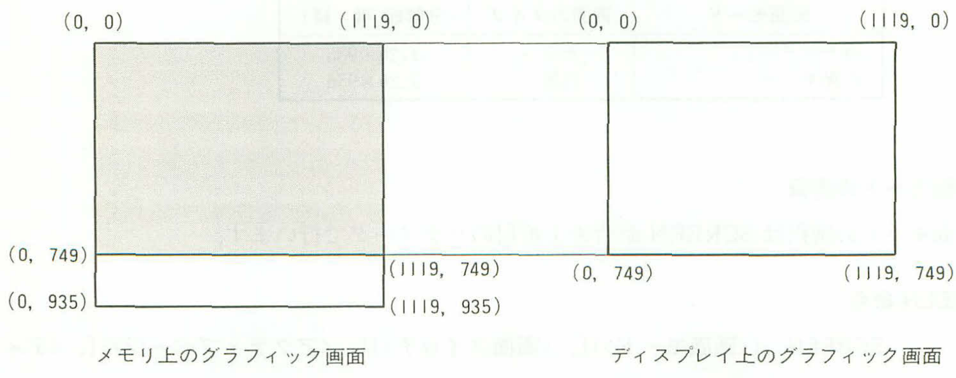
例）　カラーモードを使う場合

```
SCREEN 0
```

## ■ハイレゾモードのグラフィック画面表示についてⒽ

ハイレゾモードの場合メモリ上のグラフィック画面は1120×936ドットですが，ディスプレイ上に表示されるグラフィック画面は1120ドット×750ドットに限られます.

BASIC起動直後は，メモリのグラフィック画面の(0, 0)－(1119, 749)の範囲がディスプレイ画面上に表示されるようになっています.




この状態ではグラフィック画面の(0, 750)－(1119, 935)の範囲を見ることができません．そこで，ディスプレイ画面に表示するグラフィック画面の位置を変えるためにSCROLL命令が用意されています.

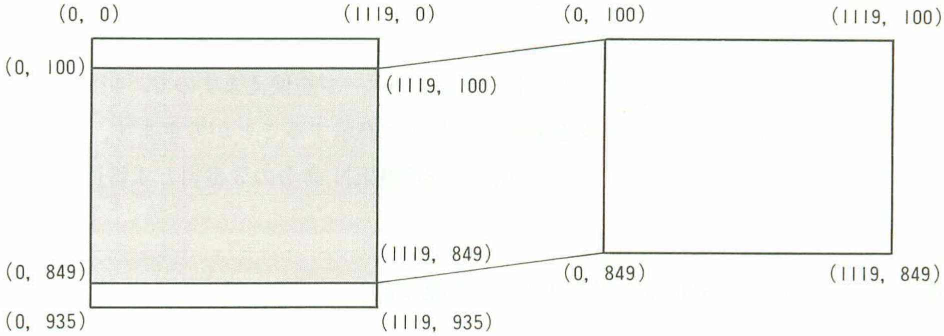
SCROLL 命令

SCROLL 〔〈数式〉〕

〈数式〉にはディスプレイ画面の表示開始Y座標を指定します．たとえば，メモリ上のグラフィック画面の(0，100)から表示したい場合は次のようにします．

SCROLL 100

このときメモリ上のグラフィック画面とディスプレイ上のグラフィック画面の関係は次のようになります．




なお，SCROLL 命令で表示位置を変更した場合，デイスプレイ上の座標が変化します．この例の場合，SCROLL 命令実行後のディスプレイ画面の左上隅の座標は(0，100)となります．

## ■ページ

BASIC の各画面モードでは，絵や図形を描いた画面を同時に何枚も保持しておき，その中から何枚かを合成して表示することができます．各モードにおける複数の画面のことをページといいます．

### ●画面モードとページ数

各画面モードにおけるページ数は次のとおりです．

ノーマルモードの場合 Ⓝ

| 画面モード | ページ数 |
|---|---|
| カラーモード | 4 |
| 白黒モード | 12(16*) |
| 高分解能白黒モード | 6(8*) |
| 高分解能カラーモード | 2 |

*のついている数値は 4096 色中•16 色モード時のページ数を表します．

▶**参照**　4096色モード→「6.3　グラフィック画面の色」

**ハイレゾモードの場合Ⓗ**

| 画面モード | ページ数 |
|---|---|
| カラーモード | 1 |
| 白黒モード | 4 |

**●アクティブページとディスプレイページ**

絵や図形を描く(書き込む)ページと画面に表示するページを別に設定することができます．描くページのことを〝アクティブページ〟，表示するページのことを〝ディスプレイページ〟といいます．

たとえば，ノーマルモードのカラーモードでは4つのページを使えますので，4ページ分の絵や図形を描いておいてそのうちの1枚を選択して次々に表示することができます．

アクティブページとディスプレイページの指定はSCREEN命令の3番目と4番目のパラメータで行います．

▶**参照**　パレットモード→「6.3　グラフィック画面の色」

**●アクティブページの指定**

〈アクティブページ〉に指定できる値と書き込むページのページ番号との対応関係は次のとおりです．〈アクティブページ〉の指定値と書き込むページとは必ず1対1に対応しています．したがって，同時に複数のページに書き込むことはできません．

**ノーマルモードの場合Ⓝ**

使用可能なページ数はパレットモードによって異なります．したがって，〈アクティブページ〉と〈ディスプレイページ〉の指定値とその意味もパレットモードによって異なりますので，ここでは区別して記述します．

BASIC起動直後はアクティブページ，ディスプレイページともページ1となっています．

**8色中・8色モードあるいは4096色中・8色モードの場合**

| 指定値 | 画面モード | 書き込むページ番号 |
|---|---|---|
| 0〜3 | カラーモード | 1〜4 |
| 0〜11 | 白黒モード | 1〜12 |
| 0〜5 | 高分解能白黒モード | 1〜6 |
| 0，1 | 高分解能カラーモード | 1，2 |

### 4096 色中・16 色モードの場合

| 指定値 | 画面モード | 書き込むページ番号 |
|---|---|---|
| 0〜3 | カラーモード | 1〜4 |
| 0〜15 | 白黒モード | 1〜16 |
| 0〜7 | 高分解能白黒モード | 1〜8 |
| 0，1 | 高分解能カラーモード | 1，2 |

### ハイレゾモードの場合 Ⓗ

| 指定値 | 画面モード | 書き込むページ番号 |
|---|---|---|
| 0 | カラーモード | 1 |
| 0〜3 | 白黒モード | 1〜4 |

### ●ディスプレイページの指定

〈ディスプレイページ〉に指定できる値と表示されるページのページ番号との対応関係は次のとおりです．なお，この表のように〈ディスプレイページ〉の指定値は表示されるページと 1 対 1 に対応しているわけではありませんので注意してください．

また，白黒モードでは複数のページを同時に表示することができます．

### ノーマルモードの場合 Ⓝ

### 8 色中・8 色モードあるいは 4096 色中・8 色モードの場合

| 指定値 | カラーモード (0) | 白黒モード (1) | 高分解能白黒モード (2) | 高分解能カラーモード (3) |
|---|---|---|---|---|
| 0 | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない |
| 1 | ページ 1 のみ表示 | ページ 1 のみ表示 | ページ 1 のみ表示 | ページ 1 のみ表示 |
| 2 | ページ 2 のみ表示 | ページ 2 のみ表示 | ページ 2 のみ表示 | × |
| 3 | × | ページ 1,2 を合成表示 | ページ 1,2 を合成表示 | × |
| 4 | × | ページ 3 のみ表示 | ページ 3 のみ表示 | × |
| 5 | × | ページ 1,3 を合成表示 | ページ 1,3 を合成表示 | × |
| 6 | × | ページ 2,3 を合成表示 | ページ 2,3 を合成表示 | × |
| 7 | × | ページ 1,2,3 を合成表示 | ページ 1,2,3 を合成表示 | × |
| 8 | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない |
| 9 | × | ページ 4 のみ表示 | × | × |
| 10 | × | ページ 5 のみ表示 | × | × |
| 11 | × | ページ 4,5 を合成表示 | × | × |
| 12 | × | ページ 6 のみ表示 | × | × |
| 13 | × | ページ 4,6 を合成表示 | × | × |
| 14 | × | ページ 5,6 を合成表示 | × | × |

| 指定値 | カラーモード (0) | 白黒モード (1) | 高分解能白黒モード (2) | 高分解能カラーモード (3) |
|---|---|---|---|---|
| 15 | × | ページ 4,5,6 を合成表示 | × | × |
| 16 | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない |
| 17 | ページ 3 のみ表示 | ページ 7 のみ表示 | ページ 4 のみ表示 | ページ 2 のみ表示 |
| 18 | ページ 4 のみ表示 | ページ 8 のみ表示 | ページ 5 のみ表示 | × |
| 19 | × | ページ 7,8 を合成表示 | ページ 4,5 を合成表示 | × |
| 20 | × | ページ 9 のみ表示 | ページ 6 のみ表示 | × |
| 21 | × | ページ 7,9 を合成表示 | ページ 4,6 を合成表示 | × |
| 22 | × | ページ 8,9 を合成表示 | ページ 5,6 を合成表示 | × |
| 23 | × | ページ 7,8,9 を合成表示 | ページ 4,5,6 を合成表示 | × |
| 24 | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない |
| 25 | × | ページ 10 のみ表示 | × | × |
| 26 | × | ページ 11 のみ表示 | × | × |
| 27 | × | ページ 10,11 を合成表示 | × | × |
| 28 | × | ページ 12 のみ表示 | × | × |
| 29 | × | ページ 10,12 を合成表示 | × | × |
| 30 | × | ページ 11,12 を合成表示 | × | × |
| 31 | × | ページ 10,11,12 を合成表示 | × | × |

×印指定不可

## 4096色中・16色モードの場合

| 指定値 | カラーモード (0) | 白黒モード (1) | 高分解能白黒モード (2) | 高分解能カラーモード (3) |
|---|---|---|---|---|
| 0 | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない |
| 1 | ページ 1 のみ表示 | ページ 1 のみ表示 | ページ 1 のみ表示 | ページ 1 のみ表示 |
| 2 | ページ 2 のみ表示 | ページ 2 のみ表示 | ページ 2 のみ表示 | × |
| 3 | × | ページ 1,2 を合成表示 | ページ 1,2 を合成表示 | × |
| 4 | × | ページ 3 のみ表示 | ページ 3 のみ表示 | × |
| 5 | × | ページ 1,3 を合成表示 | ページ 1,3 を合成表示 | × |
| 6 | × | ページ 2,3 を合成表示 | ページ 2,3 を合成表示 | × |
| 7 | × | ページ 1,2,3 を合成表示 | ページ 1,2,3 を合成表示 | × |
| 8 | × | ページ 4 のみ表示 | ページ 4 のみ表示 | × |
| 9 | × | ページ 1,4 を合成表示 | ページ 1,4 を合成表示 | × |
| 10 | × | ページ 2,4 を合成表示 | ページ 2,4 を合成表示 | × |
| 11 | × | ページ 1,2,4 を合成表示 | ページ 1,2,4 を合成表示 | × |
| 12 | × | ページ 3,4 を合成表示 | ページ 3,4 を合成表示 | × |
| 13 | × | ページ 1,3,4 を合成表示 | ページ 1,3.4 を合成表示 | × |
| 14 | × | ページ 2,3,4 を合成表示 | ページ 2,3,4 を合成表示 | × |
| 15 | × | ページ 1,2,3,4 を合成表示 | ページ 1,2,3,4 を合成表示 | × |
| 16 | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない |

| 指定値 | カラーモード (0) | 白黒モード (1) | 高分解能白黒モード (2) | 高分解能カラーモード (3) |
|---|---|---|---|---|
| 17 | × | ページ5のみ表示 | × | × |
| 18 | × | ページ6のみ表示 | × | × |
| 19 | × | ページ5,6を合成表示 | × | × |
| 20 | × | ページ7のみ表示 | × | × |
| 21 | × | ページ5,7を合成表示 | × | × |
| 22 | × | ページ6,7を合成表示 | × | × |
| 23 | × | ページ5,6,7を合成表示 | × | × |
| 24 | × | ページ8のみ表示 | × | × |
| 25 | × | ページ5,8を合成表示 | × | × |
| 26 | × | ページ6,8を合成表示 | × | × |
| 27 | × | ページ5,6,8を合成表示 | × | × |
| 28 | × | ページ7,8を合成表示 | × | × |
| 29 | × | ページ5,7,8を合成表示 | × | × |
| 30 | × | ページ6,7,8を合成表示 | × | × |
| 31 | × | ページ5,6,7,8を合成表示 | × | × |
| 32 | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない |
| 33 | ページ3のみ表示 | ページ9のみ表示 | ページ5のみ表示 | ページ2のみ表示 |
| 34 | ページ4のみ表示 | ページ10のみ表示 | ページ6のみ表示 | × |
| 35 | × | ページ9,10を合成表示 | ページ5,6を合成表示 | × |
| 36 | × | ページ11のみ表示 | ページ7のみ表示 | × |
| 37 | × | ページ9,11を合成表示 | ページ5,7を合成表示 | × |
| 38 | × | ページ10,11を合成表示 | ページ6,7を合成表示 | × |
| 39 | × | ページ9,10,11を合成表示 | ページ5,6,7を合成表示 | × |
| 40 | × | ページ12のみ表示 | ページ8のみ表示 | × |
| 41 | × | ページ9,12を合成表示 | ページ5,8を合成表示 | × |
| 42 | × | ページ10,12を合成表示 | ページ6,8を合成表示 | × |
| 43 | × | ページ9,10,12を合成表示 | ページ5,6,8を合成表示 | × |
| 44 | × | ページ11,12を合成表示 | ページ7,8を合成表示 | × |
| 45 | × | ページ9,11,12を合成表示 | ページ5,7,8を合成表示 | × |
| 46 | × | ページ10,11,12を合成表示 | ページ6,7,8を合成表示 | × |
| 47 | × | ページ9,10,11,12を合成表示 | ページ5,6,7,8,を合成表示 | × |
| 48 | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない | 全ページ表示しない |
| 49 | × | ページ13のみ表示 | × | × |
| 50 | × | ページ14のみ表示 | × | × |
| 51 | × | ページ13,14を合成表示 | × | × |
| 52 | × | ページ15のみ表示 | × | × |
| 53 | × | ページ13,15を合成表示 | × | × |
| 54 | × | ページ14,15を合成表示 | × | × |
| 55 | × | ページ13,14,15を合成表示 | × | × |
| 56 | × | ページ16のみ表示 | × | × |

| 指定値 | カラーモード (0) | 白黒モード (1) | 高分解能白黒モード (2) | 高分解能カラーモード (3) |
|---|---|---|---|---|
| 57 | × | ページ 13,16 を合成表示 | × | × |
| 58 | × | ページ 14,16 を合成表示 | × | × |
| 59 | × | ページ 13,14,16 を合成表示 | × | × |
| 60 | × | ページ 15,16 を合成表示 | × | × |
| 61 | × | ページ 13,15,16 を合成表示 | × | × |
| 62 | × | ページ 14,15,16 を合成表示 | × | × |
| 63 | × | ページ 13,14,15,16 を合成表示 | × | × |

×印指定不可

例 1)　高分解能カラーモードでページ 1 にグラフィック命令を実行しながら表示させる場合.

```
SCREEN  3, , 0, 1
```

例 2)　白黒モードでページ 3 にグラフィック命令を実行しながらページ 1，2 を表示させる場合.

```
SCREEN  1, , 2, 3
```

この場合，実行されているグラフィック命令の結果は画面には現れない．例 3)で初めて現れる．

例 3)　ページ 3 を表示する場合.

```
SCREEN  , , , 4
```

ハイレゾモードの場合 Ⓗ

| 指定値 | カラーモード (0) | 白黒モード (1) |
|---|---|---|
| 0 | 全ページ表示しない | 全ページ表示しない |
| 1 | ページ 1 を表示 | ページ 1 のみ表示 |
| 2 | × | ページ 2 のみ表示 |
| 3 | × | ページ 1, 2 を合成表示 |
| 4 | × | ページ 3 のみ表示 |
| 5 | × | ページ 1, 3 を合成表示 |
| 6 | × | ページ 2, 3 を合成表示 |
| 7 | × | ページ 1, 2, 3 を合成表示 |
| 8 | × | ページ 4 のみ表示 |
| 9 | × | ページ 1, 4 を合成表示 |
| 10 | × | ページ 2, 4 を合成表示 |
| 11 | × | ページ 1, 2, 4 を合成表示 |
| 12 | × | ページ 3, 4 を合成表示 |
| 13 | × | ページ 1, 3, 4 を合成表示 |
| 14 | × | ページ 2, 3, 4 を合成表示 |
| 15 | × | ページ 1, 2, 3, 4 を合成表示 |

×印指定不可

**例 1）** カラーモードでページ１にグラフィック命令を実行しながら表示させる場合.

```
SCREEN  0, , 0, 1
```

**例 2）** 白黒モードでページ３にグラフィック命令を実行しながらページ１，２を表示させる場合.

```
SCREEN  1, , 2, 3
```

この場合，実行されているグラフィック命令の結果は画面には現れない．例 3)で初めて現れる.

**例 3）** ページ３を表示する場合.

```
SCREEN  , , , 4
```

## ■画面スイッチ

画面スイッチとは，グラフィック画面を一時的に消去するスイッチです．SCREEN 命令の２番目のパラメータによって設定します．画面スイッチに指定する値とグラフィック画面の表示の関係は次のとおりです．BASIC 起動直後には表示を行うようになっています．

| 指定値 | グラフィック画面の表示 |
|---|---|
| 0または1 | 表示を行う |
| 2または3 | 現在の表示を一時的に消去する |

これはプログラムの作成中にグラフィック画面を一時的に消してプログラムのテキストを見やすくする場合などに便利です.

## 5.3 画面ハードコピー

ディスプレイに表示された画面のイメージをそのままプリンタで印刷することを〝ハードコピー〟といいます．

BASICには，各種のハードコピー機能が提供されています．

### ■画面ハードコピー機能の種類

BASICでは次の4つのハードコピー機能を使用することができます．

| ハードコピー機能の種類 | 対応プリンタ | ノーマルモード | ハイレゾモード |
|---|---|---|---|
| 通常の画面ハードコピー機能 | 各種プリンタ | ○ | ○ |
| カラー画面ハードコピー機能 | PC-PR201V系カラープリンタ | ○ | × |
| 拡大/縮小コピーおよび印字方向設定機能 | PC-PR601系ページプリンタ | ○ | △ |
| $N_{88}$-日本語BASIC(86)と同等の画面ハードコピー機能 | 各種プリンタ | ○ | × |

○：使用可　△：一部機能使用不可　×：使用不可

なお，どの機能を使用した場合でも，テキスト画面あるいはグラフィック画面だけのハードコピーを出力させたり，テキスト画面とグラフィック画面を合成して出力することができます．

### ■通常の画面ハードコピー機能

通常の画面ハードコピー機能は次のとおりです．

| キー操作 | COPY命令 | 機　能 |
|---|---|---|
| CTRL + COPY | COPY　1 | テキスト画面のみを出力<br>書体はプリンタの印字体 |
| GRPH + COPY | COPY　2 | グラフィック画面のみを出力 |
| COPY | COPY　3 | テキスト画面とグラフィック画面を合成して出力<br>テキストの書体は画面のドットイメージ |

## ■カラー画面ハードコピー機能

PC-PR201V 系のカラープリンタが必要です．

| キー操作 | COPY 命令 | 機　能 |
|---|---|---|
| CTRL + COPY | COPY 1 | テキスト画面のみをカラーで出力<br>書体はプリンタの印字体 |
| GRPH + COPY | COPY 2 | グラフィック画面のみをカラーで出力<br>白と黒は反転する |
| COPY | COPY 3 | テキスト画面とグラフィック画面を合成してカラーで出力<br>白と黒は反転する<br>テキストの書体は画面のドットイメージ |
| なし | COPY 4 | グラフィック画面のみをカラーで出力<br>白と黒は反転しない |
| なし | COPY 5 | テキスト画面とグラフィック画面を合成してカラーで出力<br>白と黒は反転しない<br>テキストの書体は画面のドットイメージ |

---

**注意**：カラーコピーは8色中•8色モードでのみ使用可能です．4096色中•8色，4096色中•16色モードではモノクロのハードコピーになります．

---

## ■拡大／縮小コピーおよび印字方向の設定機能

PC-PR601 系のページプリンタが必要です．

| キー操作 | COPY 命令 | 機　能 | 印字方向 | 印字倍率 |
|---|---|---|---|---|
| CTRL + COPY | COPY 1 | テキスト画面のみを出力<br>書体はプリンタの印字体 | 水平または垂直 | — |
| GRPH + COPY | COPY 2 | グラフィック画面のみを出力 | 水平または垂直 | * |
| COPY | COPY 3 | テキスト画面とグラフィック画面を合成して出力<br>テキストの書体は画面のドットイメージ | 水平または垂直 | * |

*：標準，1/3倍，2/3倍，3/3倍，4/3倍のうちいずれか．Ⓝ
　標準，2/3倍，4/3倍のうちのいずれか．Ⓗ

印字方向の欄の〝水平〟は用紙の走行方向が画面の上下に対応していることを意味します．また，〝垂直〟は用紙の走行方向が画面の左右に対応していることを意味します．

印字倍率の欄の〝標準〟とは PC-PR201 系のプリンタで出力される大きさを表します．

---

**注意**：縮小/拡大の設定(標準以外)および,印字方向の設定(垂直)を PC-PR601 系以外のプリンタで行うと画面ハードコピーは正しく動作しません．

---

## ■ $N_{88}$-日本語 BASIC(86)と同等の画面ハードコピー機能

$N_{88}$-日本語 BASIC(86)の「通常の画面ハードコピー機能」と同等の機能です.

| キー操作 | COPY 命令 | 機　能 |
|---|---|---|
| CTRL + COPY | COPY　1 | テキスト画面のみを出力<br>書体はプリンタの印字体 |
| GRPH + COPY | COPY　2 | グラフィック画面のみを出力 |
| COPY | COPY　3 | テキスト画面とグラフィック画面を合成して出力<br>テキストの書体はプリンタの印字体 |
| なし | COPY　4 | グラフィック画面を縦方向に縮小して出力(640×200 ドットのモードのみ) |
| なし | COPY　5 | テキスト画面とグラフィック画面を縦方向に縮小して出力.<br>テキストの書体はプリンタの印字体(640×200 ドットのモードのみ) |

## ■画面ハードコピー機能の設定方法

各画面ハードコピー機能の設定はMS-DOSのSWITCHコマンドとセットアップユーティリティ〝N88SETUP〟を使って行います.

| 機　能 | ユーティリティの設定 | | | | 対応プリンタ |
|---|---|---|---|---|---|
| | MS-DOS の SWITCH | N88SETUP | | | |
| | プリンタの設定 | 機能 | 方向 | 倍率 | |
| 通常の画面ハードコピー機能 | 24 ドットまたは 16 ドット | モノクロコピー | 水平 | 標準 | 各種プリンタ |
| カラー画面ハードコピー機能 | 24 ドット | カラーコピー | 水平 | 標準 | PC-PR201V 系カラープリンタ |
| 縮小/拡大コピーおよび印字方向設定機能 | 24 ドット | モノクロコピー | それぞれ可 | それぞれ可 | PC-PR601 系ページプリンタ |
| $N_{88}$-日本語 BASIC(86)と同等の画面ハードコピー機能 | 24 ドットまたは 16 ドット | $N_{88}$-BASIC 互換 | —— | —— | 各種プリンタ |

PC-PR201系, PC-PR201V系, PC-PR601系などのプリンタを使用する場合は, MS-DOSのSWITCHコマンドでプリンタの設定を24ドット系プリンタの使用に設定してください. その他の16ドット系プリンタを使用する場合は16ドット系プリンタの使用に設定してください.

▶参照　セットアップユーティリティ→「12.2　セットアップユーティリティ」
SWITCH コマンド→「MS-DOS のマニュアル」

# 第6章

# グラフィックス

BASICにはグラフィックスを扱うための豊富な命令が用意されています．これらの命令を組み合わせて使用することにより，さまざまな線，図形などを描くことができます．この章ではグラフィックスを扱う上で知っておかなければならない次のような重要な概念を説明します．

・グラフィック画面の座標系

・パレット機能

・タイリング

・画像の転送

第5章でみたようにBASICにはグラフィックスに関わるモードとしてノーマルモードとハイレゾモードがあります．2つのモードはグラフィック画面の解像度が異なるため，座標に関わる命令では値の指定のしかたなどが異なりますが，3つの座標系，パレット機能，タイリング，画像の転送などの考え方や命令の概要は共通です．

この章では，ノーマルモードの場合とハイレゾモードの場合を別に説明しています．ノーマルモードの例としては高解像度カラーモードの場合を中心に解説しています．また，特に断り書きがない場合は，両モードに共通な内容を記述しています．

なお，個々のグラフィック命令についての詳細は「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」を参照してください．

## 6.1　グラフィック画面の座標系

グラフィック画面に図形を表示する際には，画面のどの位置に出力するかを指定するために〝座標〟という概念を用います．この座標は数学などで用いられる座標と同じ考え方のもので，2つの直交する座標軸の原点からの距離を指定することによって平面上の1点を決めるものです．

BASICのグラフィック画面には，オリジナルスクリーン座標系，スクリーン座標系，ワールド座標系の3つの座標系が用意されています．この座標系をうまく使うことによって複雑な図形を描いたり，図形の移動，拡大，縮小などを用意に行うことができます．

グラフィック画面の３つの座標系

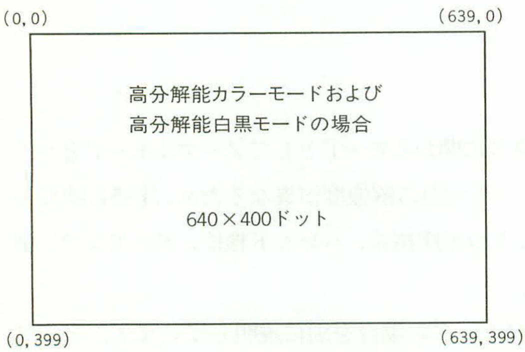
グラフィック画面の座標系
- ワールド座標系
- スクリーン座標系
- オリジナルスクリーン座標系

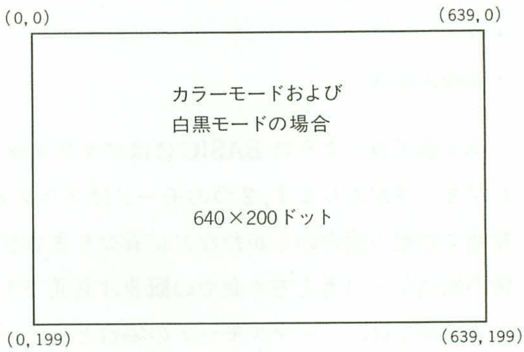
## ■オリジナルスクリーン座標系

オリジナルスクリーン座標系はディスプレイ装置と結びついた座標系です．この座標系の大きさはディスプレイ装置と対応しており，座標上の各点が画面を分解するドット数と一致しています．

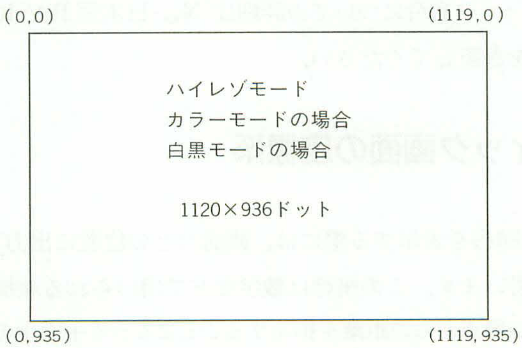
### ●オリジナルスクリーン座標系

ノーマルモードの場合 Ⓝ




ハイレゾモードの場合 Ⓗ




ハイレゾモードの場合にはオリジナルスクリーン座標系の全体を一度に表示することはできません．Y軸方向にはSCROLL命令で設定された750ドット分のみが表示されます．BASIC起動直後は(0, 0)-(1119, 749)の範囲が表示されています．

▶参照　SCROLL命令→５.２の「ハイレゾモードのグラフィック画面表示について」

## ■ビューポートとスクリーン座標系

BASICの命令を使って描いた図形を表示する際には，ディスプレイ画面の中で表示領域を限定することができます．限定した表示領域をビューポートといいます．ビューポートの設定にはVIEW命令を使います．

VIEW命令

VIEW　(Sx1, Sy1)－(Sx2, Sy2)〔,〈領域色〉〕〔,〈境界色〉〕

ここでの座標指定はディスプレイの画面に対応したオリジナルスクリーン座標系で行います．

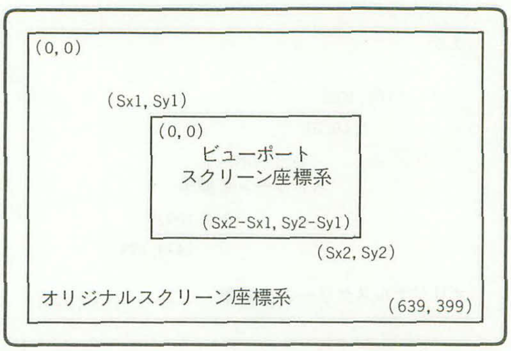
オリジナルスクリーン座標系の(Sx1, Sy1)を左上隅，(Sx2, Sy2)を右下隅とする区画が図形を表示する範囲になります．なお，Sx1<Sx2, Sy1<Sy2が成り立たない場合，あるいはこれらの座標がオリジナルスクリーン座標から外れている場合にはエラーになります．

ビューポート内部の座標系をスクリーン座標系といいます．スクリーン座標系は，ビューポートの左上隅を原点(0, 0)としてドット単位に座標をとり直したものです．

領域色にパレット番号を指定するとビューポートの内部がその色でぬりつぶされます．境界色にパレット番号を指定するとビューポートの枠がその色で描かれます．

●VIEW命令の図解

ノーマルモード(高分解能カラーモード)の場合Ⓝ



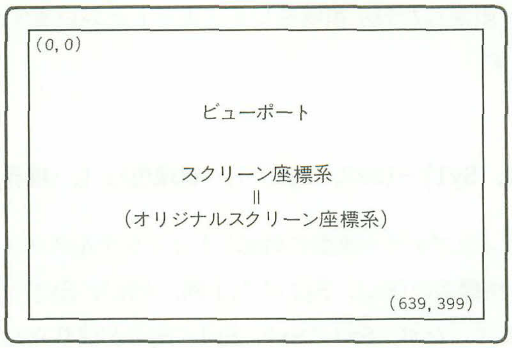

ハイレゾモードの場合Ⓗ

オリジナルスクリーン座標の右下隅の座標は(1119, 935)です．

●VIEW命令を実行しない初期状態のビューポート

オリジナルスクリーン座標の画面全体がビューポートになっています．したがって，スクリーン座標系＝オリジナルスクリーン座標系となります．

ノーマルモード(高分解能カラーモード)の場合 Ⓝ



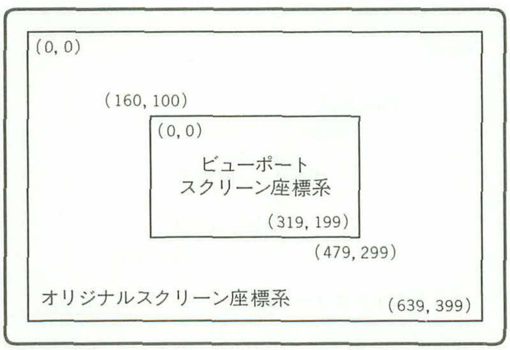

ハイレゾモードの場合 Ⓗ

オリジナルスクリーン座標系の右下隅の座標は(1119, 935)です.

●ディスプレイ画面にビューボードを設定した場合の例

ノーマルモード(高分解能カラーモード)の場合 Ⓝ

VIEW (160, 100)－(479, 299)を実行すると, ビューポートは画面の中央になります. スクリーン座標は(0, 0)～(319, 199)の範囲に新しく設定されます.




ハイレゾモードの場合 Ⓗ

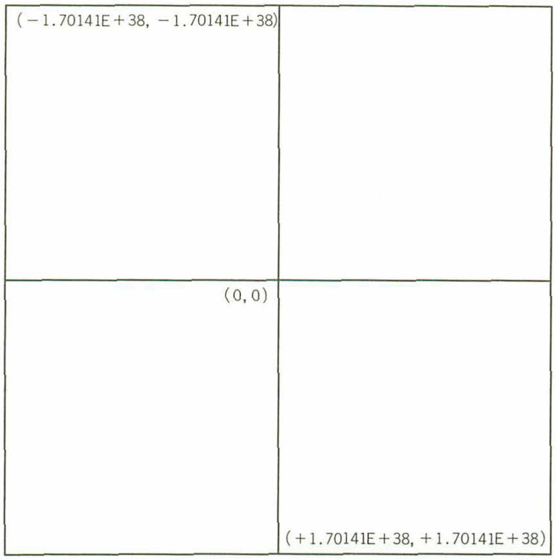
同じ命令を実行すると画面の左上寄りにビューボードが設定されます.

## ■ワールド座標系

大部分の描画コマンドはこの座標系を使用して描画を行います. この座標系はディスプレイ画面に対応したものではなくいわば論理的な座標です. ワールド座標系の大きさは縦横とも－1.70141E＋38～＋1.70141E＋38 の範囲まで許されます. ユーザーはこの範囲の領域に PSET,

DRAW，CIRCLE，LINE などのグラフィック命令を使って絵を描くことができます．ワールド座標系を利用することによりスクリーンの物理的な大きさに制限されない広いワールド座標に，線を引いたりグラフや図を描いたりすることができます．

### ●ワールド座標系




ワールド座標系はノーマルモードの場合も，ハイレゾモードの場合も同じです．

## ■ウィンドウ

BASIC が設定した論理上の座標系であるワールド座標系は，範囲がたいへん広いのでワールド座標系のすべての1ドット1ドットをディスプレイのドットに対応させて表示することはできません．そこで，ワールド座標に描かれた図形を画面に表示させるためには，ワールド座標上の任意の範囲を，実際のディスプレイ上の表示領域(ビューポート)に割り当てます．この任意の領域のことをウィンドウと呼びます．

このウィンドウの設定を行うのが WINDOW 命令です．

### WINDOW 命令

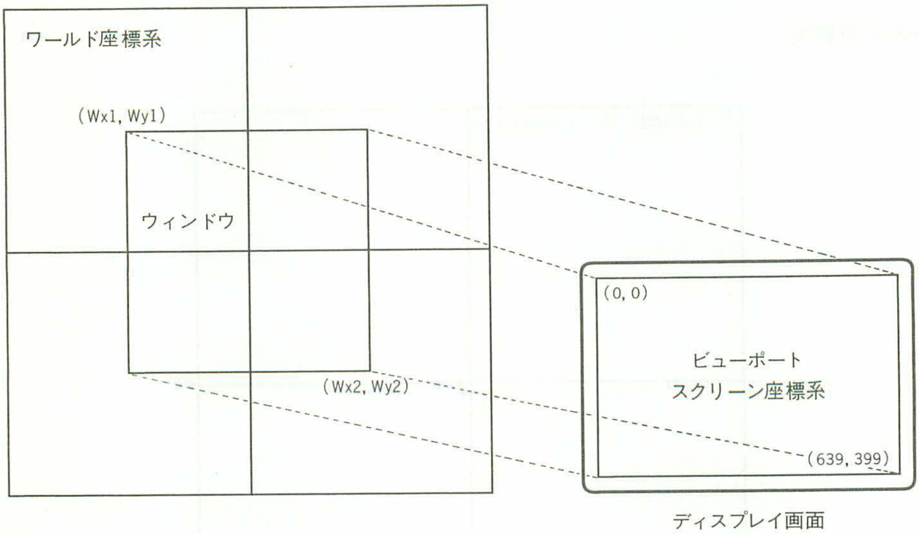
WINDOW　(Wx1，Wy1)−(Wx2，Wy2)

ワールド座標上の(Wx1，Wy1)を左上隅，(Wx2，Wy2)を右下隅とする範囲がウィンドウになります．座標の指定はワールド座標で行います．Wx1＜Wx2，Wy1＜Wy2 が成り立たない場合，あるいはこれらの座標がワールド座標系から外れている場合にはエラーになります．

この例はビューポートは初期状態でオリジナルスクリーン座標の画面全体となっています．

● WINDOW 命令の図解

ノーマルモード(高分解能カラーモード)の場合 Ⓝ



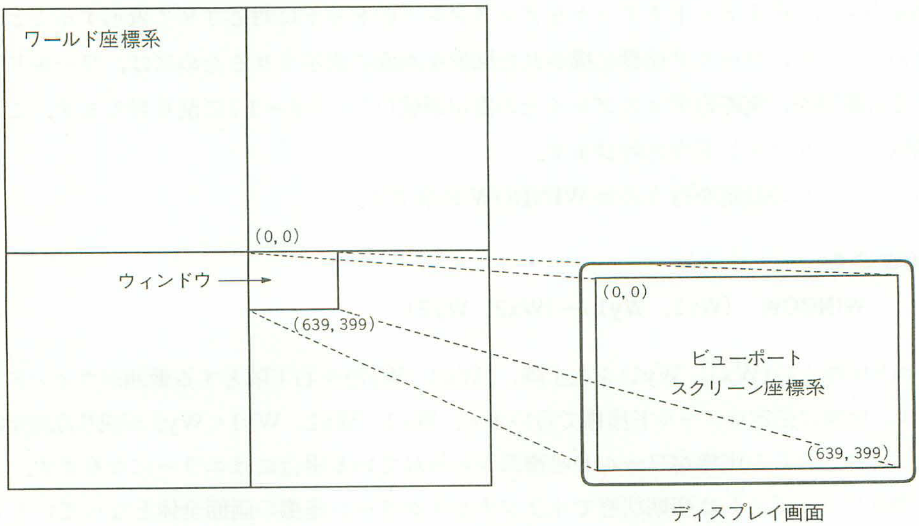

ハイレゾモードの場合 Ⓗ

オリジナルスクリーン座標系の右下の座標は(1119, 935)です.

● WINDOW 命令を実行しない初期状態のウィンドウ

ノーマルモード(高分解能カラーモード)の場合 Ⓝ

ワールド座標系の(0, 0)-(639, 399)がスクリーン座標系の(0, 0)-(639, 399)に対応しています.



ハイレゾモードの場合 Ⓗ

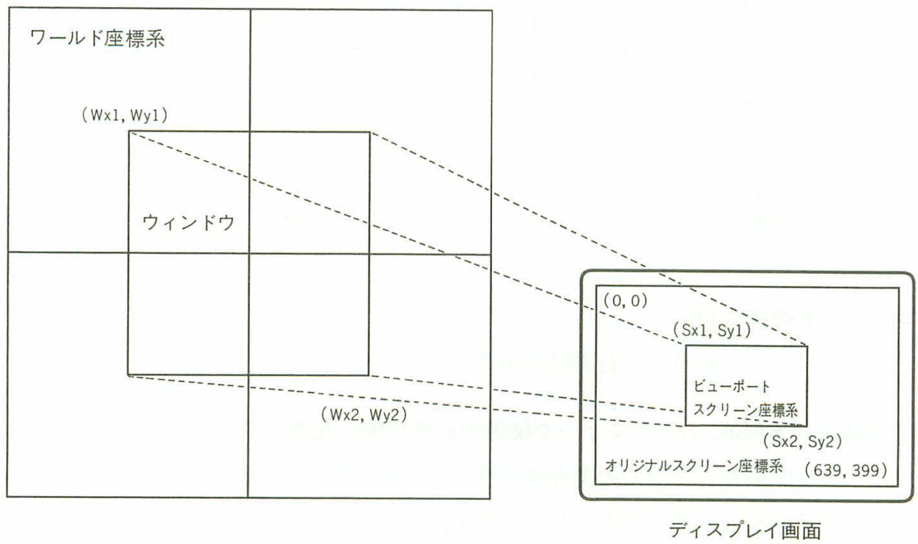
ワールド座標系の(0, 0)-(1119, 935)がスクリーン座標系の(0, 0)-(1119, 935)に対応しています.

● WINDOW 命令と VIEW 命令を両方実行した場合

ノーマルモード(高分解能カラーモード)の場合 Ⓝ

WINDOW　(Wx1, Wy1)－(Wx2, Wy2)

VIEW　(Sx1, Sy1)－(Sx2, Sy2)




ハイレゾモードの場合 Ⓗ

オリジナルスクリーン座標の右下隅の座標が(1119, 935)です.

## ■ 3 つの座標系の関係

オリジナルスクリーン座標系, スクリーン座標系, ワールド座標系の 3 つの座標系とウィンドウ, ビューポートの関係をみてみましょう.

### ● 3 つの座標系の例

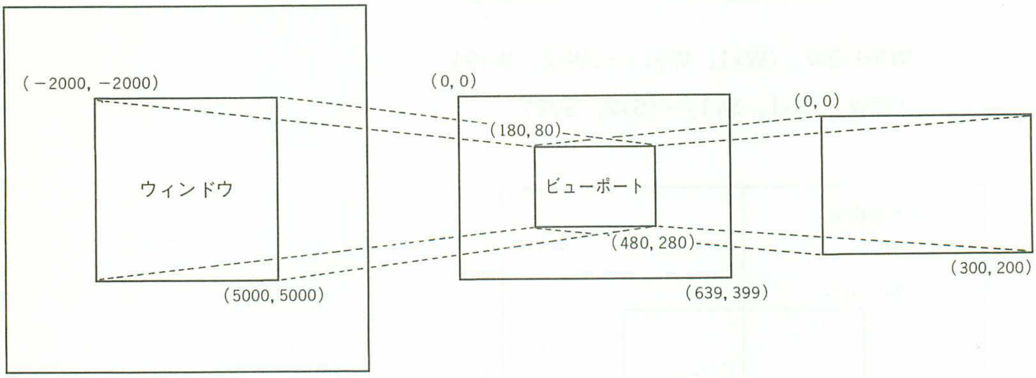
次のような命令を実行します.

WINDOW (－2000, －2000)－(5000, 5000)

VIEW (180, 80)－(480, 280)

ワールド座標系のなかの(−2000, −2000)−(5000, 5000)をウィンドウとして取り出します．オリジナルスクリーン座標系の(180, 80)−(480, 280)をビューポートとします．これでウィンドウ内の図形がビューポート内に表示されます．

**ノーマルモード(高分解カラーモード)の場合 Ⓝ**




**ハイレゾモードの場合 Ⓗ**

オリジナルスクリーン座標系の右下隅の座標は(1119, 935)です．

**● 3つの座標系とBASICのグラフィック関連の命令で使う座標**

グラフィックに関係する命令の大部分は座標の指定が必要です．次の表はそれぞれの命令がどの座標系を使うのかということを示しています．

| 座標系 | グラフィック関連の命令・関数 |
|---|---|
| ワールド座標系 | CIRCLE, DRAW, LINE, PAINT, POINT, PRESET, PSET, WINDOW |
| オリジナルスクリーン座標系 | VIEW, SCROLL* |
| スクリーン座標系 | GET@, PUT@, POINT(関数) |

* SCROLLはハイレゾモードのみ使用することができます．Ⓗ

各命令についての詳細は「N₈₈-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」を参照してください．

# 6.2 ウィンドウとビューポート

ワールド座標に描いた図形は，ウィンドウやビューポートの設定によってディスプレイ画面内での位置や大きさが変わってきます．

実際にワールド座標に図形を描いた上でWINDOW命令とVIEW命令を使って画面の変化を見てみます．

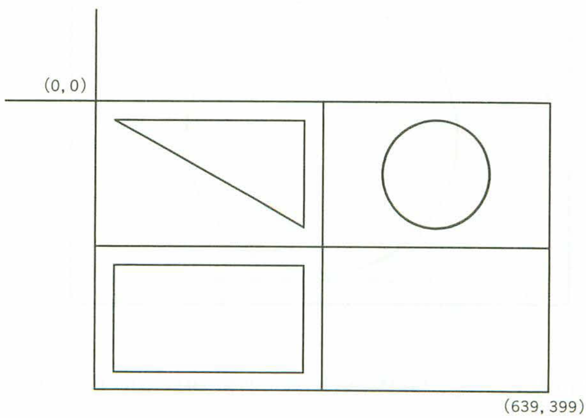
この項のプログラム例はノーマルモードの高解像度カラーモードの場合(SCREEN 3を実行した状態)です．ハイレゾモードで実行する場合はSCREEN命令のパラメータを変更してください(SCREEN　0)．同様に実行することができます．ただし，ノーマルモードの場合とはディスプレイ上での表示位置が異なります．

## ■ワールド座標上の図形

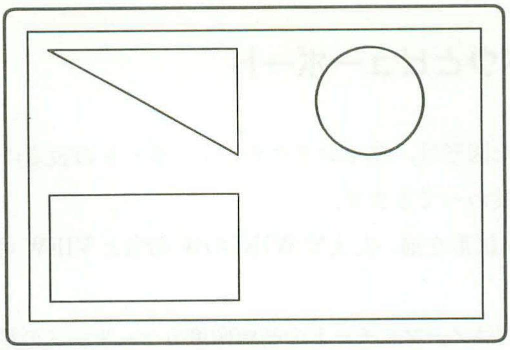
まず，次のプログラムを実行します．

```
10 SCREEN 3 :CLS 3
40 CIRCLE (479,100),90,6
50 LINE(20,40)-(199,40),1:LINE -(199,159),1
60 LINE(20,40)-(199,159),1
70 LINE(20,219)-(199,379),2,B
```

ワールド座標上に次のような状態で描かれます．




ディスプレイ画面でもこのように見えます．初期状態ではウィンドウはワールド座標系の(0，0)－(639，399)であり，スクリーン座標系の(0，0)－(639，399)に対応しているのです．

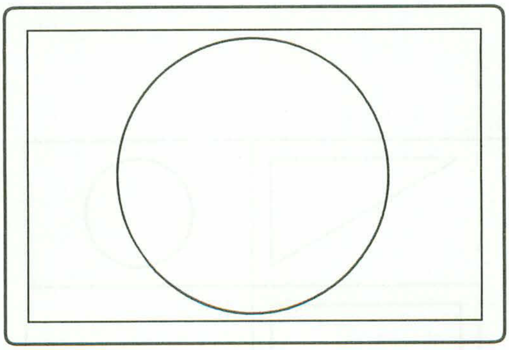


■ウィンドウの設定

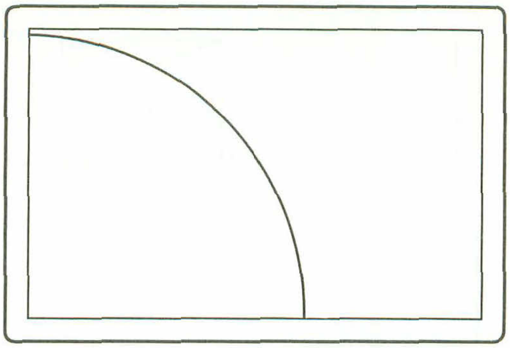
画面の右上の円の描かれている部分だけを表示するために，始めのプログラムの前に，WINDOW命令を入れてみます．

```
10 SCREEN 3 :CLS 3
20 WINDOW (319,0)-(639,199)
40 CIRCLE (479,100),90,6
50 LINE(20,40)-(199,40),1:LINE -(199,159),1
60 LINE(20,40)-(199,159),1
70 LINE(20,219)-(199,379),2,B
```

このプログラムを実行すると画面は次のようになります．

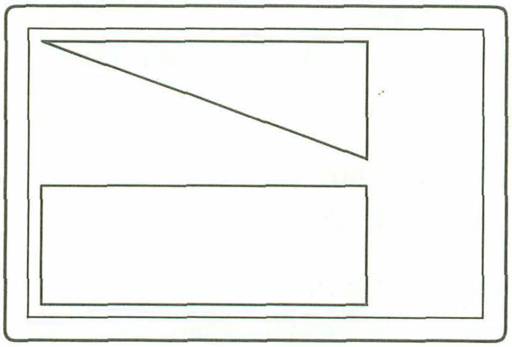


次に20行目のWINDOW命令を〝20 WINDOW (479, 0)－(639, 100)″に変えてみます．



次に三角形と四角形の部分を画面全体に表示してみましょう．

20行目のWINDOW命令を〝20 WINDOW (0，0)－(319，399)〟に変えて実行します．

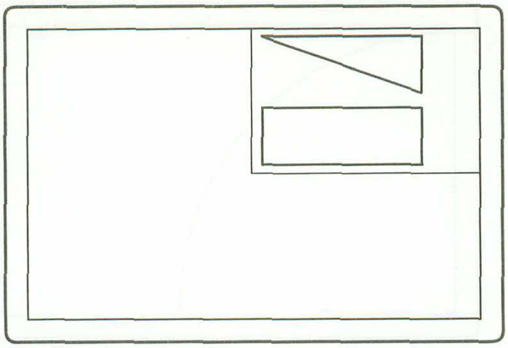


このように，WINDOW命令を使うとワールド座標に描かれた図形を任意に拡大，縮小したり移動させたりできるようになります．

## ■ビューポートの設定

ワールド座標に描かれた図形をディスプレイの右上に表示させるには，VIEW命令を使ってビューポートを画面の右上に設定します．それには前に作ったプログラムにVIEW命令を追加しWINDOW命令を変更します．

```
10 SCREEN 3 :CLS 3
20 WINDOW (0,0)-(319,399)
30 VIEW (319,0)-(639,199),,4
40 CIRCLE (479,100),90,6
50 LINE(20,40)-(199,40),1:LINE -(199,159),1
60 LINE(20,40)-(199,159),1
70 LINE(20,219)-(199,379),2,B
```



## ■ワールド座標値の制限

CIRCLE命令やLINE命令などのグラフィック命令に使用するワールド座標の座標値には，次のような最小値と最大値の制限があります．最小値以下あるいは最大値以上の値を指定するとオバーフローのエラーが起こります．

この最小値と最大値はいつも一定ではありません．WINDOW命令とVIEW命令で指定した座標値によって変化します．

```
WINDOW  (a1, b1)-(a2, b2)
VIEW  (x1, y1)-(x2, y2)
```

各座標にこのような値を指定してWINDOW命令とVIEW命令を実行した場合には，各種グラフィック命令で使用できるワールド座標値の範囲は，次のような式で表すことができます．

WXはワールド座標系のX座標の値を，WYワールド座標系のY座標の値を表します．

WX

$$(-32768-x1)\frac{a2-a1}{x2-x1}+a1 \leqq WX \leqq (32767-x1)\frac{a2-a1}{x2-x1}+a1$$

WY

$$(-32768-y1)\frac{b2-b1}{y2-y1}+b1 \leqq WY \leqq (32767-y1)\frac{b2-b1}{y2-y1}+b1$$

たとえば，

```
WINDOW  (0, 0)-(639, 399)
VIEW  (0, 0)-(639, 399)
```

の場合を計算してみると，

−32768≦WX≦32767

−32768≦WY≦32767

となります．

---

**注意**：VIEW 命令を使う前には，必ず WINDOW 命令を使ってウィンドウを設定してください．VIEW 命令だけを単独で使うとウィンドウが変化してしまいます．
また，SCREEN 命令は WINDOW 命令と VIEW 命令の前に実行してください．SCREEN 命令を実行すると，ビューポートとウインドウの設定は初期化されます．

---

# 6.3 グラフィック画面の色

## ■グラフィック画面の色指定

グラフィック画面ではバックグラウンドカラー(背景色)，ボーダーカラー(周辺色)，フォアグラウンドカラー(前景色)を指定することができます．

グラフィック画面の色指定には〔1〕COLOR 命令を使います．

### 〔1〕COLOR 命令

COLOR 〔<ファンクションコード>〕〔，<バックグラウンドカラー>〕〔，<ボーダーカラー>〕〔，<フォアグラウンドカラー>〕〔，<パレットモード>〕

この中でファンクションコードはテキスト画面の文字にいろいろな機能を与えるものです．

**ハイレゾモードの場合 Ⓗ**

<パレットモード>の指定は意味を持ちません．

▶参照　ファンクションコード→「5.1 テキスト画面」

<ボーダーカラー>はバックグラウンドの周辺の色を表します．BASIC 起動直後は黒です．

| 指定値 | 色 |
|---|---|
| 0 | 黒 |
| 1 | 青 |
| 2 | 赤 |
| 3 | 紫 |
| 4 | 緑 |
| 5 | 水色 |
| 6 | 黄 |
| 7 | 白 |

ただし，〈ボーダーカラー〉は，ノーマルモードで400ラインの高解像度ディスプレイの使用時およびハイレゾモードでは意味がありません．

〈バックグラウンドカラー〉はグラフィック画面の地の色を表します．

〈フォアグラウンドカラー〉はグラフィック画面に点や線を表示したりするときに使われる色を表します．CIRCLE，LINE，PSET命令などの描画命令で色の指定を省略するとこの色が採用されます．

なお，〈バックグラウンドカラー〉と〈フォアグラウンドカラー〉の指定は色番号で指定するわけではありません．後で説明する〝パレット番号〟で指定します．

パレット番号については次の項を参照してください．

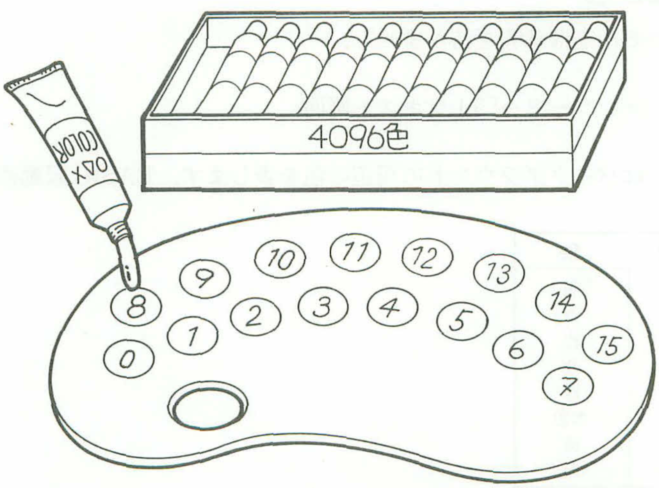
## ■パレットモード

BASICでは〔1〕COLOR命令，PAINT命令，LINE命令などグラフィック画面の色指定を行う際には，固定した色を指定するのではなく，パレットと呼ばれる〝色の入れ物〟の番号を指定します．パレット番号にはユーザーが自由に色を割り当てることができます．各命令で表示色にパレット番号を指定するとそのパレット番号に割り当てられている色が使用されるというわけです．これを〝パレット機能〟といいます．パレット機能の利点は，いったん描いてしまった図形でも，後からパレット番号に割り当てられた色(カラーコード)を変更するだけで図形の色を変えることが可能な点です．

カラーの画面モードでは表示可能な色の種類が異なります．

### ハイレゾモードの場合 Ⓗ

4096色中•16色モードのみを使用することができます．このため〔1〕COLOR命令での〈パレットモード〉の指定は意味を持ちません．



**8色中･8色モード**

黒，明るい青，明るい赤，明るい紫，明るい緑，明るい水色，明るい黄，白の8色を使用することができます．

**4096色中･8色モード**

4096色の中から8色を使用することができます．

**4096色中･16色モード**

4096色の中から16色を使用することができます．

この3つのパレットモードのどれを選択するかは〔1〕COLOR命令の5番目のパラメータで設定します．ノーマルモードでのBASICの起動直後は8色中・8色モードになっています．

| 指定値 | パレットモード |
|---|---|
| 0 | 8色中･8色モード |
| 1 | 4096色中･8色モード |
| 2 | 4096色中･16色モード |

## ■ノーマルモード時の基本グラフィックモードと拡張グラフィックモード

BASICのグラフィックには2つのグラフィックモードがあります．これはパレットモードの選択を可能にするかどうかを決めるものです．

| | グラフィック機能の差 |
|---|---|
| 基本グラフィックモード | 8色中･8色モードだけを使用できる |
| 拡張グラフィックモード | 8色中･8色モード<br>4096色中･8色モード<br>4096色中･16色モード<br>の3つのモードが使用可能 |

この章のノーマルモードの解説は拡張グラフィックモードについて行っています．基本グラフィックモードの機能は拡張グラフィックモードの8色中･8色モードと同等です．

拡張グラフィックモードにするためにはディップスイッチSW1の8番をONにしてMS-DOSのシステムを起動します．出荷時にはONに設定してありますが確認してください．

---

**注意**：MS-DOS(Ver 3.3以降)でグラフィックスドライバを使用している場合は，ディップスイッチの設定に関係なく拡張グラフィックモードになります．

PC-9801/E/F/Mでは拡張グラフィックモードは使用できません．

---

**▶参照**　ディップスイッチ→本体添付の「ガイドブック」

グラフィックスドライバ→「MS-DOSのマニュアル」

## ■パレットの色の変更

パレットの色の変更は〔2〕COLOR 命令で行います.

### 〔2〕COLOR 命令

1)　COLOR〔=(〈パレット番号〉, 〈カラーコード〉)〕

2)　COLOR =( |B|W| , 〈濃淡の階調〉) Ⓗ

1)はノーマルモードおよびハイレゾモードの, カラーの画面モードの場合に使用することができます.

カラーコードとは各色につけられたコード番号です. この命令ではパレット番号とカラーコードを対応づけることによって, どのパレットにどの色を対応させるか任意に決めることができます. その際, 同じカラーコードの色を複数のパレット番号に対応づけてもかまいません.

2)はハイレゾモードの白黒の画面モードの場合にのみ使用できます. 白黒の濃淡を表現することができます. 詳しくはハイレゾモードの場合の解説の部分で説明しています.

ノーマルモードの白黒の画面モードの場合には〔2〕COLOR 命令を使ってパレットの色を指定することはできません. この場合〔1〕COLOR 命令, PAINT, LINE 命令などで指定されたパレット番号が 0 のときは黒, 0 以外の場合は白で色がぬられます.

### ノーマルモードおよびハイレゾモードの, カラーの画面モードの場合

カラーの画面モードにおけるパレット番号とカラーコードの関係は次のとおりです.

| パレット番号 | カラーコード | | |
|---|---|---|---|
| | 8色中・8色モード | 4096色中・8色モード | 4096色中・16色モード |
| 0 | 0(黒) | &H000 | &H000 |
| 1 | 1(明るい青) | | |
| 2 | 2(明るい赤) | | |
| 3 | 3(明るい紫) | | |
| 4 | 4(明るい緑) | | |
| 5 | 5(明るい水色) | | |
| 6 | 6(明るい黄) | | |
| 7 | 7(白) | &HFFF | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |
| 14 | | | |
| 15 | | | &HFFF |

8色中・8色モード Ⓝ

0～7のパレット番号に8個のカラーコードを任意に指定します。

4096色・8色モード Ⓝ

0～7のパレット番号に&H000～&HFFFの4096個のカラーコードから任意に指定します。

4096色・16色モード ⓃⒽ

0～15のパレット番号に&H000～&HFFFの4096個のカラーコードから任意に指定します。

たとえば，4096色中・16色モードで

```
COLOR=(0, &H777)
```

とすると，パレット番号0が灰色となります。

パレットをBASICが起動した時の初期状態にもどすには，〝COLOR ⏎〃とだけ入力します。ただし，ノーマルモードの基本グラフィックモードではこの命令でパレットを初期化することはできません。

BASIC起動直後の各パレットには次のカラーコードが設定されています。

| パレット番号 | カラーコード | | | 色 |
|---|---|---|---|---|
| | 8色中・8色モード | 4096色中・8色モード | 4096色中・16色モード | |
| 0 | 0 | &H000 | &H000 | 黒 |
| 1 | 1 | &H00F | &H00F | 明るい青 |
| 2 | 2 | &H0F0 | &H0F0 | 明るい赤 |
| 3 | 3 | &H0FF | &H0FF | 明るい紫 |
| 4 | 4 | &HF00 | &HF00 | 明るい緑 |
| 5 | 5 | &HF0F | &HF0F | 明るい水色 |
| 6 | 6 | &HFF0 | &HFF0 | 明るい黄 |
| 7 | 7 | &HFFF | &HFFF | 白 |
| 8 | | | &H777 | 灰色 |
| 9 | | | &H00A | 少し暗い青 |
| 10 | | | &H0A0 | 少し暗い赤 |
| 11 | | | &H0AA | 少し暗い紫 |
| 12 | | | &HA00 | 少し暗い緑 |
| 13 | | | &HA0A | 少し暗い水色 |
| 14 | | | &HAA0 | 少し暗い黄 |
| 15 | | | &HAAA | 少し暗い白 |

パレット設定後は，グラフィック画面に表示されている色およびこれから画面に対して行うすべてのグラフィック命令(PSET，LINEなど)の指定色は，そのパレット番号に設定されているカラーコードの色で表示されます。

▶**参照**　カラーコードと色の関係→6.3の「色の仕組み(カラーコードの指定方法)」

### ハイレゾモードの白黒の画面モードの場合 Ⓗ

白黒モードでは，〔2〕COLOR命令を使って次のような書式でパレットの濃淡を指定し，濃い灰色や淡い灰色などを表示することができます．

```
COLOR =( |B|, 〈濃淡の階調〉)
         |W|
```

黒パレット(B)と白パレット(W)の2つのパレットが使用できます．この2つのパレットに16段階の濃淡を設定することができます．〈濃淡の階調〉に指定する値と濃淡の関係は次のようになります．

| 〈濃淡の階調〉の指定値 | 濃淡 |
|---|---|
| &H000 | 黒 |
| &H111 | |
| &H222 | |
| &H333 | |
| &H444 | |
| &H555 | |
| &H666 | |
| &H777 | 灰色 |
| &H888 | |
| &H999 | |
| &HAAA | |
| &HBBB | |
| &HCCC | |
| &HDDD | |
| &HEEE | |
| &HFFF | 白 |

表中にない値を指定した場合，指定した値よりも小さく，その値に一番近い濃淡の階調が設定されます．

BASIC起動直後は黒パレットには&H000(黒)が，白パレットには&HFFF(白)が設定されています．

なお，パレット番号0が黒パレット，0以外が白パレットになりますので，〔1〕COLOR命令，PAINT命令，LINE命令などでパレット番号0を指定すると黒パレットの色が，0以外を指定すると白パレットの色がぬられます．

---

**注意**：パレットモードによって各画面モードでの使用可能なページ数も変わってきます．ページ数に関しては5.2の「ページ」を参考にしてください．

なお，ノーマルモードの4096色中•8色モードと4096色中•16色モードはアナログRGB対応ディスプレイに対応したモードです．アナログRGBディスプレイ以外のディスプレイを接続している場合は，このモードで指定どおりの色を出すことはできません．

---

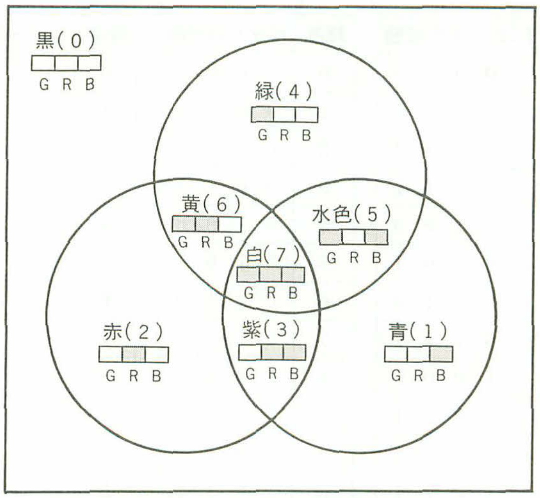
### ■色の仕組み（カラーコードの指定方法）

カラーの画面モードにおいてはグラフィック画面に表示できる色はすべて基本色である緑(G)，赤(R)，青(B)の組み合せで表現されています．

#### 8色中・8色モードの場合 Ⓝ

緑(G)，赤(R)，青(B)のそれぞれがONまたはOFFの2とおりを選択できます．したがって，2×2×2=8色が表現できます．

R，G，B各色のON／OFFとカラーコードの関係は次のようになります．




□はONの状態を，□はOFFの状態を示しています．

#### 4096色中・8色モードと4096色中・16色モードの場合 Ⓝ

緑(G)，赤(R)，青(B)のそれぞれを16段階の輝度（光の強さ）で表現することができます．したがって，16×16×16=4096色が使用可能です．

R，G，B各色の輝度は4ビットの情報を持っています．これを1桁の16進数にすると，R，G，B各色の輝度は0～Fまでの16段階の数値で表すことができます．

カラーコードは16進表記で3桁の数値で表されていますが，この3桁の数字がR，G，B各色の輝度がこの数値に対応しているわけです．なお，カラーコードの順番はG，R，Bの順になっています．

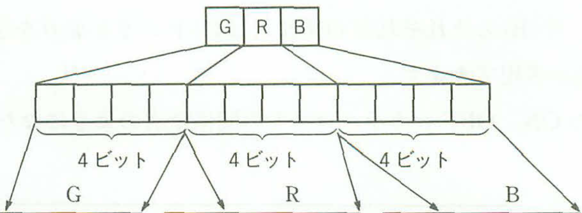
| R, G, B の輝度 |
|---|
| Gの輝度が&H9<br>Rの輝度が&HF<br>Bの輝度が&H0 |

⇨

| カラーコード | 色 |
|---|---|
| &H9F0 | オレンジ色 |

たとえば，Gが&H9(暗い緑)でRが&HF(最も明るい赤)の場合にはオレンジ色になります．このような方法で4096色をカラーコードで表現しているのです．

R，G，B各色の輝度とカラーコードの関係は次のようになります．




G

| 輝度 | ビットの状態 |
|---|---|
| 0 | 0 0 0 0 |
| 1 | 0 0 0 1 |
| 2 | 0 0 1 0 |
| 3 | 0 0 1 1 |
| 4 | 0 1 0 0 |
| 5 | 0 1 0 1 |
| 6 | 0 1 1 0 |
| 7 | 0 1 1 1 |
| 8 | 1 0 0 0 |
| 9 | 1 0 0 1 |
| A | 1 0 1 0 |
| B | 1 0 1 1 |
| C | 1 1 0 0 |
| D | 1 1 0 1 |
| E | 1 1 1 0 |
| F | 1 1 1 1 |

R

| 輝度 | ビットの状態 |
|---|---|
| 0 | 0 0 0 0 |
| 1 | 0 0 0 1 |
| 2 | 0 0 1 0 |
| 3 | 0 0 1 1 |
| 4 | 0 1 0 0 |
| 5 | 0 1 0 1 |
| 6 | 0 1 1 0 |
| 7 | 0 1 1 1 |
| 8 | 1 0 0 0 |
| 9 | 1 0 0 1 |
| A | 1 0 1 0 |
| B | 1 0 1 1 |
| C | 1 1 0 0 |
| D | 1 1 0 1 |
| E | 1 1 1 0 |
| F | 1 1 1 1 |

B

| 輝度 | ビットの状態 |
|---|---|
| 0 | 0 0 0 0 |
| 1 | 0 0 0 1 |
| 2 | 0 0 1 0 |
| 3 | 0 0 1 1 |
| 4 | 0 1 0 0 |
| 5 | 0 1 0 1 |
| 6 | 0 1 1 0 |
| 7 | 0 1 1 1 |
| 8 | 1 0 0 0 |
| 9 | 1 0 0 1 |
| A | 1 0 1 0 |
| B | 1 0 1 1 |
| C | 1 1 0 0 |
| D | 1 1 0 1 |
| E | 1 1 1 0 |
| F | 1 1 1 1 |

暗い ↓ 明るい

輝度0は各色が表示されないことを表す．

### ハイレゾモードの場合 Ⓗ

ノーマルモードの場合の4096色中・16色モードと同様になります．

## ■タイリング

BASICの描画命令を使用して任意の範囲をぬりつぶすことができます．この場合，パレット番号を使用して単色でぬりつぶすのが一般的ですが，複数の色を使ったモザイク模様(タイルパターン)でぬりつぶすことも可能です．これを〝タイリング〟と呼びます．

ノーマルモードの場合も，ハイレゾモードの場合もタイリングの方法は全く同様です．

白黒の画面モードや8色中･8色モードで中間色を表現するためのテクニックとして特に効果的です．また，4096色中･8色モードや4096色中･16色モードでも使用できます．

この機能は〔2〕PAINT 命令や CIRCLE 命令，LINE 命令などで使うことができます．パレット番号を指定する代わりにタイルパターンを表す文字列を指定します．この文字列を〝タイルストリング〟といいます．

タイルパターンの大きさは横方向は8ドット分と決められていますが，縦方向の長さはタイルストリングの文字列の長さで決まります．縦方向がnドットのタイルを作るためには，タイルストリングとして，白黒の画面モードでn文字，8色中•8色モード，4096色中•8色モードで3×n文字，4096色中•16色モードで4×n文字の長さがそれぞれ必要です．

タイルパターンは，タイルストリングに指定する文字列のキャラクタコードを2進数で表現したもの(ビットパターン)によって決定されます．タイルストリングの指定方法は白黒の画面モード，8色モード，16色モードによって異なります．

### ●白黒の画面モード

タイルストリング中の各1文字(バイト)のコードがタイルの横8ドットに対応します．文字コードの2進数表現で，タイルの横8ドットのうち，1に対応するドットはセット(白)，0に対応するドットはリセット(黒)として表されます．〈タイルストリング〉がn文字の長さであれば，その文字列の表す模様は，このようにして決定される横8ドットのパターンを縦にn列分だけ並べたパターンになります．

例） ノーマルモードの白黒モードの場合

CHR$(&HAA)+CHR$(&H55)

| ドット | 1 2 3 4 5 6 7 8 |
|---|---|
| &HAA | 1 0 1 0 1 0 1 0 |
| 対応する色 | 白黒白黒白黒白黒 |
| &H55 | 0 1 0 1 0 1 0 1 |
| 対応する色 | 黒白黒白黒白黒白 |

この例では，タイルストリングが2文字ですから横8ドットのタイルパターンが2列分設定されます．この2列のタイルパターンは1であるビットと0であるビットが互いに逆になっています．このパターンを基本タイルとして指定された領域を埋めれば，細かい市松模様になりますが，スクリーンのドットがたいへん細かいため，灰色のように見えます．

### ●カラーの画面モード

白黒の画面モードと同様に，タイルパターンはタイルストリングに対応するドットパターンによって決定されます．白黒の画面モードと異なる点は，8色中•8色モード，4096色中•8色モードの場合は3文字分ごとに，また4096色中•16色モードの場合は4文字分ごとに，タイルの横8ドット(1列分)の色がパレット番号として設定されることです．タイルパターンの例をみてみましょう．

例)　8色中・8色モード，4096色中・8色モードの場合

CHR$(&HAA)+CHR$(&H55)+CHR$(&HFF)+CHR$(&H55)+CHR$(&HAA)+CHR$(&HFF)

| ドット | 1 2 3 4 5 6 7 8 | |
|---|---|---|
| &HAA | 1 0 1 0 1 0 1 0 | ← $2^0$の桁 |
| &H55 | 0 1 0 1 0 1 0 1 | ← $2^1$の桁 |
| &HFF | 1 1 1 1 1 1 1 1 | ← $2^2$の桁 |
| 対応するパレット番号 | 5 6 5 6 5 6 5 6 | |
| &H55 | 0 1 0 1 0 1 0 1 | ← $2^0$の桁 |
| &HAA | 1 0 1 0 1 0 1 0 | ← $2^1$の桁 |
| &HFF | 1 1 1 1 1 1 1 1 | ← $2^2$の桁 |
| 対応するパレット番号 | 6 5 6 5 6 5 6 5 | |

各ドットについての3ビット分のデータがパレット番号に読み換えられます．この場合，指定するタイルストリングのうちの最初の文字が最下位ビット($2^0$)の桁を表し，以後，$2^1$の桁，$2^2$の桁となります．

この例ではタイルストリングが6文字ですから横8ドットのタイルパターン2列分が設定されていることになります．この2列のタイルパターンは両方ともパレット番号5とパレット番号6が交互に横8ドット並びますが，色の順序が1ドットずれているのでパレットが初期状態であれば水色と黄の市松模様になります．実際にはドットが細かいので淡い黄緑に見えます．

例)　4096色中・16色モードの場合

CHR$(&HCE)+CHR$(&H8C)+CHR$(&HD0)+CHR$(&H22)

| ドット | 1 2 3 4 5 6 7 8 | |
|---|---|---|
| &HCE | 1 1 0 0 1 1 1 0 | ← $2^0$の桁 |
| &H8C | 1 0 0 0 1 1 0 0 | ← $2^1$の桁 |
| &HD0 | 1 1 0 1 0 0 0 0 | ← $2^2$の桁 |
| &H22 | 0 0 1 0 0 0 1 0 | ← $2^3$の桁 |
| 対応するパレット番号 | 7 5 8 4 3 3 9 0 | |

この例ではタイルストリングが4文字ですから横8ドットのタイルパターンが1列分設定されているわけです．各ドットについての4ビット分のデータがパレット番号に読み換えられることにより，このような結果となります．

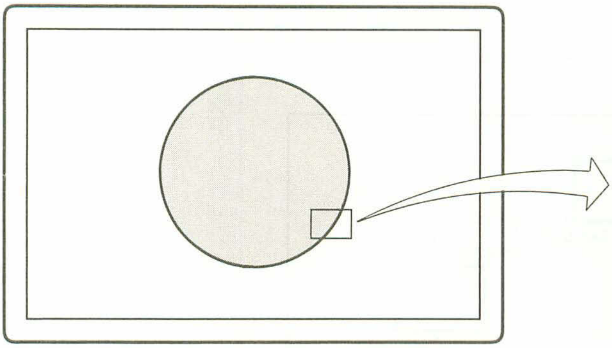
パレットが初期状態の場合には，横8ドットが白，明るい水色，灰色，明るい緑，明るい紫，明るい紫，少し暗い青，黒と並ぶタイルパターンになります．

実際にタイリングを使った描画例を示します．

4096色中・16色モードの場合の例です．

```
10 SCREEN 3 : COLOR ,,,,2
20 CLS 2
30 TILE1$=CHR$(&H0)+CHR$(&HFF)+CHR$(&H55)+CHR$(&H0)
40 TILE2$=CHR$(&H0)+CHR$(&HFF)+CHR$(&HAA)+CHR$(&H0)
50 CIRCLE (320,200),100,7
60 PAINT (320,200),TILE1$+TILE2$,7
```

TILE1＄がタイルパターンの1列目をTILE2＄が2列目を表しています．この2列のタイルパターンは両方ともパレット番号2とパレット番号6が交互に横8ドット並びますが，色の順序が1ドットずれているのでパレットが初期状態であれば赤と黄の市松模様になります．実際にはドットが細かいので白で描かれた円の中はオレンジ色に見えます．



各ドットのパレット番号

| 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 7 | 0 | 0 |
| 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 7 | 0 | 0 |
| 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 7 | 0 | 0 | 0 |
| 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 7 | 0 | 0 | 0 |
| 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 7 | 0 | 0 | 0 | 0 |
| 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 7 | 0 | 0 | 0 | 0 | 0 |
| 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 7 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 2 | 6 | 2 | 6 | 2 | 6 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 6 | 2 | 6 | 2 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

このように，タイリングを行うとタイルストリングで設定されたタイルパターンを単位として，領域がぬりつぶされます．

**注意**：カラーの画面モードの場合，タイルストリングの文字数が3(4096色中•16色モードの場合は4)の倍数でないときには，文字列の後ろから3(あるいは4)で割った余りの数だけ無視されます．また，3文字(あるいは4文字)に満たない場合には "Illegal function call" エラーとなります．

## 6.4 図形データの複写

BASICでは画面上のある大きさの領域をそっくりそのまま他の場所へ複写させることができます．これを行うためにはGET@命令とPUT@命令を組み合わせて使用します．

PUT@命令とGET@命令の書式は次のとおりです．

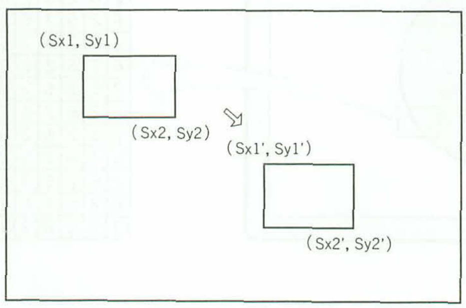
GET@命令

GET〔@〕(Sx1, Sy1)－(Sx2, Sy2), 〈配列変数名〉〔(〈添字〉)〕

PUT@命令

PUT〔@〕(Sx1, Sy1), 〈配列変数名〉〔(〈添字〉)〕〔, 〈条件〉〕〔, 〈フォアグラウンドカラー〉, 〈バックグラウンドカラー〉〕

▶参照　GET@, PUT@命令→「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」




この例のように(Sx1, Sy1)と(Sx2, Sy2)で指定される領域に表示されている図形を(Sx1', Sy1')と(Sx2', Sy2')で指定される領域に複写することができます．(Sx2', Sy2')はもとの図形の大きさと同じになるように自動的に決まります．

GET@命令は画像データを配列に読み込み，PUT@命令で配列のデータを画像に変えて出力します．

使用する配列変数はGET@命令を使う前にDIM命令を使って定義しておかなくてはなりません．確保すべき配列変数の大きさは，次のようにして求めることができます．

①まず，読み込みたい図形の縦横のドット数を数え，必要なバイト数を算出します．

**必要なバイト数＝((横のドット数＋7)¥8)＊縦のドット数＊M＋4**

ただし，Mは画面モードによって変わります．

白黒の画面モードのとき…………………………M＝1

8色中・8色，4096色中・8色モードのとき ……M＝3

4096色中・16色モードのとき……………………M＝4

②次に，必要なバイト数から実際に確保する配列変数の大きさ(添字の値)を決めます．

**添字の値＝(必要なバイト数＋1)¥N－1**

ただし，Nは配列変数の型によって変わります．

整数型配列……………N＝2

単精度型配列…………N＝4

倍精度型配列…………N＝8

③求めた添字の値をもとに，配列変数(例：A％)を宣言します．

**DIM　A％(添字の値)**

なお，配列変数(ここではA％)にはOPTION BASE(添字の最小値)が0の1次元配列を使用するようにしてください．

PUT@命令の〈条件〉は複写先に表示する際のいろいろな条件を指定するものです．そのまま表示するにはPSETを使います．

また，GET@命令とPUT@命令を使う際にアクティブページを変えると別のページに複写することも可能です．

```
10 SCREEN 1
20 I=(((100+7)¥8)*100*1+4+1)¥2-1
30 DIM A%(I)
40 SCREEN ,,0
50 CIRCLE (50,50),20,1
60 GET@ (0,0)-(99,99),A%
70 SCREEN ,,1
80 PUT@(100,100),A%,PSET
90 SCREEN ,,,2
```

これは白黒モードの場合の例です．

この場合にはページ1の(0，0)～(99，99)の範囲の図形がページ2の(100，100)を左上の頂点とする位置へ複写されます．

# 第7章

# 日本語処理

BASICには日本語処理機能が用意されています．この機能を使用することにより，日本語文字列(2バイト系文字列)を英数カナ文字列(1バイト系文字列)同様にテキスト画面に表示したり，漢字プリンタに印字したりすることができます．

## 7.1　日本語キャラクタモードとグラフィックキャラクタモード

BASICは日本語の内部コードとしてシフトJISコードを使用しています．

▶**参照**　日本語内部コード→「7.4　日本語文字列の内部形式」

シフトJISコードは日本語1文字を2バイトで表現するように工夫されたものですが，シフトJISの第1バイト目が，キャラクタコードのグラフィック文字のコードと重なっています．このため，シフトJISコードとグラフィックキャラクタコードを混在して使用することはできません．

---

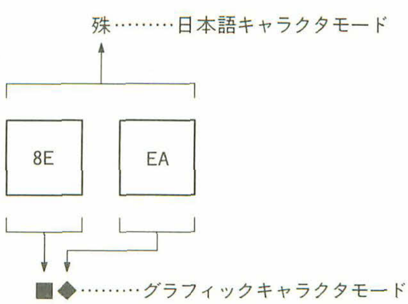
**参考**：シフトJISコードの第1バイト目は&H81～&H9F，&HE0～&HFCです．

---

たとえば，

```
PRINT  CHR$(&H8E)+CHR$(&HEA)
```

を実行した場合，グラフィックキャラクタモードと日本語キャラクタモードとでは次のように異なる文字が表示されます．



この2つのモード(キャラクタモード)の選択はCONSOLE命令の〈キャラクタモード〉で行い,0ならば日本語キャラクタモード,1ならばグラフィックキャラクタモードとなります.BASICの起動直後は日本語キャラクタモードとなっています.

CONSOLE命令

CONSOLE　〔〈スクロール開始行〉〕〔,〈スクロール行数〉〕〔,〈ファンクションキー表示スイッチ〉〕〔,〈カラー／白黒スイッチ〉〕〔,〈キャラクタモード〉〕

▶参照　CONSOLE命令→「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」

・日本語キャラクタモード

日本語文字の入力(INPUT命令など),表示(PRINT命令など),印刷(LPRINT命令など)および各種日本語命令・関数の使用が可能です.

グラフィックキャラクタの入力,表示,印刷はできません.もし,実行すると日本語やグラフィックキャラクタ以外の文字まで正しく出力できない場合があります.

・グラフィックキャラクタモード

グラフィックキャラクタの入力,表示,印刷が可能です.

日本語の入力,表示,印刷および各種日本語命令・関数の使用はできません.もし,実行すると日本語やグラフィックキャラクタ以外の文字まで正しく出力できない場合があります.

## 7.2 日本語入力の方法

ダイレクトモードで画面上のプログラムの編集などを行っているときや，プログラムモードでINPUT命令などを実行したときのように，キーボードからキー入力が可能な状態にあるとき，[CTRL]キーを押しながら[XFER]キーを押すと日本語入力モードとなります．日本語入力モードではAI逐次変換，AI連文節変換，逐次変換，連文節変換，単文節変換などの入力方式による日本語文字(漢字など)の入力が可能になります．このモードを解除するには再び[CTRL]キーを押しながら[XFER]キーを押すか，[STOP]キーまたは[⏎]キーを押します．

また，KINPUT命令を実行すると，自動的に日本語入力モードになります．KINPUT命令による日本語入力モードを解除するには，[⏎]キーを押すか，または[STOP]キーを押します．KINPUT命令の場合，[CTRL]キーを押しながら[XFER]キーを押しても日本語入力モードは解除されませんので注意してください．

BASICにおける日本語入力機能は，内部的にはMS-DOSが提供する日本語入力機能を使用して実現されています．したがって，日本語の入力方式は，MS-DOSの設定の場合と同じとなります．このため，設定によってはJIS16進コード入力に限定される場合があります．

---

**注意**：市販の日本語入力(処理)用フロントエンドプロセッサ(日本語FEP)を使用する場合，次のような点に注意してください．

- 日本語入力モードへ入る／日本語入力モードから出るときの操作キーが異なることがある．
- 日本語入力モードでのキーの働きや画面の表示は，日本語FEPが提供する機能にしたがう．
- KINPUT命令は通常のINPUT命令と同様の動作になる．

各日本語FEPの操作方法や機能および動作環境については，日本語FEPに添付されている説明書を参照してください．

---

▶**参照**　日本語入力の方法→「日本語入力ガイド」，「MS-DOSのマニュアル」

## 7.3　日本語文字列の扱い方

### ●文字列定数としての日本語文字列定義

文字列定数として日本語文字列を扱うときには，英数カナの1バイト系文字列と同様にダブルクォーテーション(")で囲みます.

例)「日本語」という文字列を文字変数A$に代入する.

```
A$="日本語"
```

### ●テキスト画面への表示

日本語文字列をテキスト画面に表示すると,2バイト系全角文字は1バイト文字の2文字分(2桁)の領域を占めます. 2バイト系半角文字は1バイト文字と同じ1文字分です.

日本語文字を表示するには2つの方法があります.

例1)「日本語」という文字列を文字定数として扱い，画面に表示する.

```
PRINT "日本語"
```

例2)「日本語」という文字列を文字変数A$に代入し，A$の内容を表示する.

```
A$="日本語"
PRINT A$
```

### ●グラフィック画面への表示

日本語文字をグラフィック画面へ表示するには，1文字単位でPUT@命令を使用します.

例)「漢」(JISコードで&H3441)という文字を表示する.

```
PUT@ (100, 100), KANJI(&H3441)
```

### ●日本語文字列の印刷

日本語文字列を漢字プリンタに出力すれば，日本語文字を印刷することができます.

例)「日本語」という文字をプリンタで印字する.

```
LPRINT "日本語"
```

出力される日本語の書体はプリンタが持っているフォント(文字の形)によって変わります.

### ●日本語文字列のファイル入出力

日本語文字列も1バイト系英数カナ文字列と同じようにファイルに出力することができます．また逆に，ファイルに書き込まれている日本語文字列を読み込んで変数に代入することもできます．

例）「日本語」という日本語文字列をディスクファイル(シーケンシャルファイル)に出力し，またすぐに読み出して画面に出力する．

```
100 OPEN "DATA" FOR OUTPUT AS #1
110 PRINT #1,"日本語"
120 CLOSE
130 OPEN "DATA" FOR INPUT AS #1
140 INPUT #1,A$
150 PRINT A$
160 CLOSE
```

### ●日本語文字列操作命令・関数

BASICには1バイト文字列を操作する命令・関数のほかに，日本語文字列を操作するための専用の命令・関数が用意されています．これらの命令・関数を日本語文字列操作命令・関数と呼びます．これらを使用することによって日本語文字列の操作を容易に行うことができます．

日本語文字列操作命令・関数には次のようなものがあります．

| 命令・関数 | 機　能 |
|---|---|
| AKCNV$ | 1バイト系の英数カナ文字を，対応する2バイト系全角文字に変換する |
| JIS$ | 2バイト系日本語文字のJISコード(16進表記)を得る |
| KACNV$ | 2バイト系全角文字を，対応する1バイト系の英数カナ文字に変換する |
| KEXT$ | 文字列の中から1バイト系文字だけ，あるいは2バイト系日本語文字だけのどちらかを抽出する |
| KINPUT | キーボードから入力された2バイト系日本語文字を文字変数に代入する．KINPUT命令が実行されると自動的に日本語入力モードに入る |
| KINSTR | 文字列の中から指定文字列を捜し出し，その位置を得る |
| KLEN | 2バイト系日本語文字を含む文字列中の特定タイプの文字の合計数を得る |
| KMID$ | 2バイト系日本語文字を含む文字列の中から任意の長さの文字列を抜き出す |
| KNJ$ | 指定したJISコード(16進表記)に対応する2バイト系日本語文字を得る |
| KPLOAD | 利用者定義文字パターンをシステムに登録する |
| KTYPE | 2バイト系日本語文字を含む文字列中の指定位置の文字のタイプを得る |

これらの関数を使う際，対象とする文字列には2バイト系日本語文字と1バイト系英数カナ文字の両方が含まれていてもかまいません．

**例)**「日本語」という日本語文字列の中から「本」を取り出す．

```
PRINT  KMID$("日本語", 2, 1)
本
```

▶**参照**　日本語文字列操作命令・関数→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

## 7.4 日本語文字列の内部形式

日本語文字は2バイトで表されます．したがって1バイト系英数カナ文字2文字分のメモリ(2バイト)を占有します．

BASICでは日本語をメモリに格納したり，ディスクファイルにデータとして入出力する場合，シフトJISコードを使用します．

MS-DOSの日本語内部コードもシフトJISですから，MS-DOS上で動作するアプリケーションソフトウェア，ユーティリティ，コマンドなどで作成したファイルをBASICで直接取り扱うことができます．また，その逆も可能です．

なお，シフトJISコードは，あくまでもシステムが日本語をデータとして入出力する際に使用する内部表現です．ユーザーが実際に扱う日本語入力時の日本語コードや，日本語文字列操作関数で取り扱う日本語コードなどのコードは，$N_{88}$-日本語BASIC(86)と同じJISコードです．

---

**参考**：日本語文字の内部表現であるシフトJISコードは，JISコードをある特定の法則で変換したものでJISコードと同様2バイトから構成されます．

---

## 7.5 日本語文字列を扱う場合の注意

・文字型定数の大きさは最大 255 バイトですから，すべてが英数カナ文字の場合は 255 文字が最大となります．文字列が日本語文字を含む場合は，日本語文字 1 文字を 2 バイトとして数えます．したがって，すべて日本語からなる文字型定数の最大文字数は 127 文字です．

・日本語文字を含んだ文字列の比較を行う場合，比較は 1 バイトごとにデータの 16 進表現の大きさの順番で比較されます．

・ランダムファイルで日本語文字列を扱う場合，日本語文字列の長さに注意してフィールド長を決めてください．

・PRINT　USING，LPRINT　USING 命令で日本語文字を書式制御文字列に指定したり，日本語文字を含む文字列を編集したりすることはできません．

・KEY 命令で日本語文字列をファンクションキーに登録することはできません．

・TAB 関数で日本語文字列を制御することはできません．

・日本語文字を含んだ文字列をスクリーンファイル（SCRN：）に出力しても日本語の表示は行われません．

・1 文字の日本語が 2 行にまたがると表示が乱れます．2 行にまたがらないよう表示あるいは入力を行ってください．また，すでに表示されている日本語の左右どちらかの半分に英数カナ文字（1 バイト文字）を重ねるような表示あるいは入力を行うと日本語の表示が乱れます．

・日本語文字パターンの表示（ノーマルモードの場合）Ⓝ
テキスト画面上の日本語文字の文字パターンは 16×16 ドットとなっています．画面が 25 行モードであると，各行の文字と文字がくっついて表示されます．不都合な場合には 20 行モードで表示してください．

・日本語文字パターンの表示（ハイレゾモードの場合）Ⓗ
テキスト画面上の日本語文字の文字パターンは 24×24 ドットとなっています．

## 7.6 利用者定義文字

JIS 第 1 水準，第 2 水準の文字以外に，ユーザーが任意に文字パターンを設定し，JIS コードとして登録することのできる文字があります．これを利用者定義文字といいます．文字の定義には&H7621～&H767E, &H7721～&H777E の 188 個(一部機種で 94 個)の JIS コードを使用することができます．

利用者定義文字の登録を行うと，JIS 第 1 水準，第 2 水準の漢字と同様に，JIS コードを指定することによって 2 バイトの日本語文字として使用することが可能になります．

文字パターンを登録するには，次の 2 つの方法があります．

**・BASIC による利用者定義文字の登録**

KPLOAD 命令を使って，メモリ上の利用者定義文字領域に利用者定義文字を現す文字パターンを登録します．ただしこの場合，利用者定義文字は電源を切ったときやリセットスイッチを押したときに初期化されます．

▶参照　KPLOAD 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

**・MS-DOS による利用者定義文字の登録**

MS-DOS の USKCGM コマンドを使って，利用者定義文字の文字パターンをファイルに登録しておくと，起動時に自動的に読み込まれます．

▶参照　USKCGM コマンド→「MS-DOS のマニュアル」

# 第8章 ファイル

## 8.1 ファイルとは

ファイルとは，ある一定の決まり，あるいは体系に従ってまとめられた，情報やデータのかたまり(あるいはその入れ物)のことです．たとえば，会社などで社員の個人的なデータ(人事データ)をバインダなどにとじたものならば〝人事ファイル〞，あるいは，売り上げの品目や金額などのデータをキャビネットに収めたものならば〝売り上げファイル〞，といったぐあいです．

コンピュータにおいても，このファイルの概念はまったく同じで，あるまとまりをもったデータ(あるいはその入れ物)をすべてファイルとして取り扱います．一般的なファイルとの大きな違いは，そのデータのかたまりがバインダやキャビネットではなく，フロッピィディスクやハードディスクなどに電気的に格納されることです．電気的に記録されたファイルの中のデータを取り出したり，逆に格納したりするには，コンピュータに対して手順を指示することにより行います．

BASICには，コンピュータでファイルを取り扱うために必要な命令や関数が各種用意されており，これらを組み合せて使うことによって，データをファイルの形で自由に操作することができます．

### ■ファイルの種類

ファイルはいろいろな観点から分類することができますが，BASICで一般に使用される分類法に従うと，次のようになります．

**1．内容による分類**

プログラムファイル……BASICで記述されたプログラムを記録したファイル．

データファイル…………BASICプログラムで利用するデータを記録したファイル．

**2．記録形式による分類**

アスキーファイル………人間が判読できる文字(アスキーキャラクタおよび日本語キャラクタ)から成るファイル．テキストファイルともいう．

バイナリファイル………コンピュータで利用されるコード(機械語など)から成るファイル．人間がふつうには判読できないものを指す．

**3. データファイルの形式による分類**

シーケンシャルファイル

………行単位から成るファイル．順次ファイル，逐次ファイルとも呼ばれ，アクセス(読み書き)はそのファイルの先頭部分から順に行う．

ランダムファイル………定義可能な区分(フィールド)をもつレコード単位のファイル．高速なアクセス(読み書き)が可能．

**4. 格納される装置による分類**

ディスクファイル………フロッピィディスク装置または固定ディスク装置に記録されるファイル．たんに〝ファイル〞と呼ぶ場合，このディスクファイルを指すことが多い．

デバイスファイル………いろいろな入出力装置(デバイス)に対するアクセスでやりとりされるデータを論理的に扱ったファイル．キーボードファイル，スクリーンファイル，プリンタファイル，RS-232C回線ファイルなどがある．広い意味ではディスクファイルも含めてデバイスファイルと呼ぶことがある．

このようにさまざまな分類が可能です．これらの分類はそれぞれの観点から分けており，実際のファイルはこれらの組み合せで表されます．

**例1)**　RS-232C回線から送られてくるアスキー形式のBASICプログラムのファイル

→　デバイスファイル(RS-232C回線ファイル)でシーケンシャルファイルでアスキーファイルで，かつプログラムファイル

**例2)**　ディスクにランダムファイルの形式で格納された売り上げの記録ファイル

→　ディスクファイルでランダムファイルでバイナリファイルで，かつデータファイル

## ■デバイス(入出力装置)とファイル

BASICでは，あるまとまりをもったデータのかたまり(あるいはその入れ物)をすべてファイルとして取り扱うことについては前に説明したとおりです．この概念により，ディスクに記録されたデータをファイルとして操作できるわけですが，BASICではこれをさらに一般化して，たとえばプリンタやディスプレイ画面(スクリーン)，さらにはキーボードなどもファイルとして取り扱うことができるようにしています．つまり，こうした各種の入出力装置(デバイス)を，出力データや入力データのかたまりを格納する入れ物としてみなすことにより，ファイルとして扱うようにしているわけです．

実際にBASICでデバイスに対してデータの入出力を行う場合には，そのデバイスに対応するデバイス名を指定して操作を行います．次に，デバイスと，それに対応するデバイス名を示します．〝入力／出力〟の欄には，そのデバイスに対する入力／出力の命令が使用可能であるかどうかを示してあります．

| デバイス | デバイス名 | 入力 | 出力 |
|---|---|---|---|
| ディスク | A：<br>B：<br>C：<br>〜<br>Z： | ○<br>○<br>○<br><br>○ | ○<br>○<br>○<br><br>○ |
| キーボード | KYBD： | ○ | × |
| スクリーン | SCRN： | × | ○ |
| プリンタ | LPT： | × | ○ |
| RS－232C回線 | COM1：<br>COM2：<br>COM3： | ○<br>○<br>○ | ○<br>○<br>○ |

また，各デバイスに対して使用できる入出力命令はデバイスの種類によって異なります．

| 入出力命令＼デバイス | ディスク n： | COM： | KYBD： | SCRN： | LPT： |
|---|---|---|---|---|---|
| BLOAD | ○ | ○ | × | × | × |
| BSAVE | ○ | ○ | × | ○ | ○ |
| CLOSE | ○ | ○ | ○ | ○ | ○ |
| EOF | ○ | ○ | × | × | × |
| GET | ○ | × | ○ | × | × |
| INPUT # | ○ | ○ | ○ | × | × |
| INPUT$ | ○ | ○ | ○ | × | × |
| LINE INPUT # | ○ | ○ | ○ | × | × |
| LOAD | ○ | ○ | ○ | × | × |
| LOC | ○ | ○ | ○ | × | × |
| LOF | ○ | ○ | × | × | × |
| OPEN | ○ | ○ | ○ | ○ | ○ |
| PRINT #<br>(PRINT # USING)<br>(WRITE #) | ○ | ○ | × | ○ | ○ |
| PUT | ○ | × | × | ○ | ○ |
| SAVE | ○ | ○ | × | ○ | ○ |
| WIDTH | × | ○ | × | × | ○ |
| CHAIN/MERGE | ○ | × | × | × | × |

○：使用できる　×：使用できない

## ■ファイルディスクリプタ

BASICでファイルを操作する際には，コンピュータがファイルを捜したり特定したりすることが可能なように，個々のファイルに名前をつけて取り扱います．これにより，ユーザーがその名前を指定するだけで，そのファイルに対してアクセス(読み書き)を行うことができます．この名前のことを〝ファイルディスクリプタ〟といいます．その形式は次のとおりです．

〔〈デバイス名〉〕〔〈ファイル名〉〕

〈デバイス名〉は各入出力装置につけられた名前のことで，8.1の「デバイス(入出力装置)とファイル」の項の表に示したもののうちのいずれかを指定します．〈デバイス名〉はディスク装置を対象とする場合，〈ドライブ名〉と呼ばれます．〈デバイス名〉省略時は，ディスク装置のカレントドライブとみなされます．

〈ファイル名〉はディスク装置のとき必要で，次の形式で指定します．

[¥][〈ディレクトリ名〉¥] …[〈ディレクトリ名〉¥]〈主ファイル名〉〔.〈拡張子〉〕

〈ディレクトリ名〉を省略するとカレントディレクトリとみなされます．詳しくは「MS-DOSのマニュアル」を参照してください．

〈主ファイル名〉は1～8バイトの1バイト系英数文字，カナ，日本語文字からなる文字列です．

〈拡張子〉は1～3バイトの1バイト系英数文字，カナ，日本語文字からなる文字列で，省略可能ですが，指定する場合には必ずその前にピリオド(.)をつけなければなりません．

なお，プログラムファイルで〈拡張子〉を省略したときには，〈拡張子〉として自動的に〝.BAS〟が指定されます．

MS-DOS上でファイルディスクリプタの指定を行う場合，英文字は大文字／小文字ともすべて大文字とみなされます．したがって，BASICでもファイルディスクリプタに使用される英文字はすべて大文字として扱われます．

**例)**　ディスク装置のドライブBにセットされたディスク上の〝TEST〟というファイル．

B：TEST

---

**注意**：〈デバイス名〉がキーボード(KYBD：)，スクリーン(SCRN：)およびプリンタ(LPT：)である場合は〈ファイル名〉を指定しても意味がありません．

〈デバイス名〉がRS-232C回線(COM1：～COM3：)の場合は，〈ファイル名〉に相当する場所に回線の各種パラメータなどを指定します(「8．5　デバイスファイルの扱い方」参照)．

---

## ■ファイルバッファ

BASICでファイルを扱う際は，ファイルバッファと呼ばれるメモリの領域を使用します．ファイルバッファはプログラムとファイルとの間で一時的にデータを蓄えるものです．

プログラムからファイルに対してデータを出力する，あるいはファイルからプログラムへデータを読み込むといった場合には，必ずファイルバッファを仲介します．たとえばディスクファイルにデータを書き込む場合は，いったんファイルバッファにデータを蓄え，蓄えたデータが一定単位(通常256バイト)になったら，ファイルに書き出すという動作を行っています．

つまり，ファイルに対して入出力命令を実行しても実際にはバッファに対する入出力を行っているわけで，ファイルに対する実際の入出力はバッファがいっぱい(あるいは空)になったかどうかをBASICが自動的に判別して行っています．

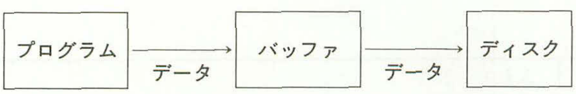
---

**参照**：ファイルバッファの読み書きの単位は次のとおりで，省略時は256バイトになります．

- インタプリタの場合，起動パラメータ/Sで指定します．詳しくは「2.3　起動パラメータの指定」を参照してください．
- コンパイラの場合，REM　$FILE命令で指定します．詳しくは第2部の「2.2　ファイル使用のため制御命令」を参照してください．

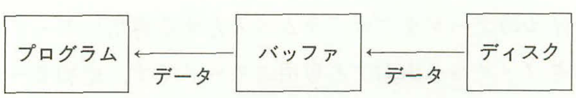
---

**プログラムからディスクへ書き込む場合**




データをバッファに書き出す．
バッファが一定単位分たまるとBASICが自動的にディスクに書き出す．

**ディスクからデータを読み込む場合**




ディスクから一定単位分のデータをバッファへ読み込む．
プログラムはバッファから必要なだけデータを読み込む．
バッファが空になったら，BASICが自動的にディスクから一定単位分読み込む．

ファイルの入出力を行う際には必ずファイルごとに専用のファイルバッファが必要です．さもないといろんなファイルのデータがファイルバッファでまじりあってしまうことになります．

ファイルとファイルバッファの対応関係は始めから決まっているものではありません．ファイルの入出力を行う前には必ず，そのファイルに専用のファイルバッファを割り当てなければなりません．

## ■ファイルのオープン／クローズ

ファイルの入出力を行う前には，前項で述べたように，使用するファイルにファイルバッファを割り当てて入出力操作が可能な状態にしなければなりません．これをファイルをオープンするといいます．また，ファイルの入出力が終了したらファイルに割り当てられていたファイルバッファを解放する必要があります．これをファイルのクローズといいます．

### ●ファイルのオープン(OPEN)

ファイルのオープンはOPEN命令を使って行います．

OPEN命令

OPEN　<ファイルディスクリプタ>　〔FOR <入出力モード>〕 AS 〔#〕<ファイル番号>

OPEN命令は，<ファイルディスクリプタ>で指定されるファイルに<ファイル番号>で表される専用バッファを割り当てます．

この命令が実行された後は，ファイルに対する入出力は<ファイル番号>を指定して行います．ファイルの入出力命令は実際にはバッファへ入出力を行っているわけですから，<ファイル番号>はファイルバッファ番号と考えてさしつかえありません．

<入出力モード>はファイルへのアクセスの仕方を指定するものです．次の4つから選択することができます．

| INPUT | 入力モード |
|---|---|
| OUTPUT | 出力モード |
| APPEND | 追加モード |
| 省略時 | ランダムモード，またはRS-232C回線ファイルの入出力 |

入力モードは既存のファイルのデータをプログラムへ入力する専用のモードです．出力モードはプログラムからデータをファイルへ出力する専用のモードです．追加モードはプログラムからデータを送り既存のファイルの終わりに追加出力する専用のモードです．

省略時はランダムモード，あるいはRS-232C回線ファイルの入出力を意味し，入力，出力の両方が可能なモードです．

---

**注意**：入力(INPUT)モード，追加(APPEND)モードでオープンした場合，指定したファイルがディスク上にないとエラーとなります．

---

例1) ドライブBのルートディレクトリ下の〝BAS〟というディレクトリ下の〝TEST〟という名前のファイルに1番のファイルバッファを割り当て，入力モードでオープンする．

```
OPEN "B:¥BAS¥TEXT" FOR INPUT AS #1
```

例2) スクリーンファイルに2番のファイルバッファを割り当て，出力モードでオープンする．

```
OPEN "SCRN:" FOR OUTPUT AS #2
```

例3) カレントドライブのカレントディレクトリの〝DATA〟という名前のファイルに3番のファイルバッファを割り当て，ランダムモードでオープンする．

```
OPEN "DATA" AS #3
```

●ファイルのクローズ(CLOSE)

ファイルの入出力が終了すると，OPEN命令で確保したファイルバッファは不要になります．そこでCLOSE命令を実行すると，ファイルに割り当てられていたファイルバッファが解放されます．

これで，解放されたファイルバッファは別のファイルのために使用することができるようになります．

また，前に述べたようにデータの出力命令は実際にはファイルバッファへのデータ出力を行うだけであり，それが実際にファイルに出力されるのはファイルバッファがいっぱいになってからです．したがって，最後にファイルバッファに出力されたデータがファイルに出力されずにファイルバッファに残ってしまう可能性があります．CLOSE命令は，このようにしてファイルバッファに残ってしまったデータを，最後に(ファイルバッファを解放する前に)ファイルに出力するという役割も持っています．

CLOSE命令

CLOSE 〔〔#〕〈ファイル番号〉〔,〔#〕〈ファイル番号〉〕…〕

CLOSE命令では〈ファイル番号〉を指定してファイルのクローズを行います．〈ファイル番号〉を省略した場合にはそのときオープンしているファイルはすべてクローズされます．

ファイルをクローズしない状態でプログラムを終了した場合には，ファイルバッファに残されたデータは失われてしまいます．このような事態を防ぐために，入出力の終わったファイルは必ずクローズするようにしてください．

なお，END命令，NEW命令，RUN命令などの実行によってもファイルがクローズされるようになっています．

## ■ファイルバッファとファイルの同時オープン数

ファイルバッファはメモリ上に確保されるものです．BASICでは0番から13番の14個のファイルバッファが確保されています．ただし，0番のファイルバッファは特殊なバッファで普通ユーザーは使用できませんから，実際には最大13個のファイルバッファが使用可能です．これは，同時に13個のファイルをオープンすることができるということを意味します．

ファイルバッファの数はBASICインタプリンタの起動時に，/F：パラメータで指定します．1〜13までの数値を指定できます．ここでの指定は，BASICが起動した後では変更することができません．変更するにはBASICを再起動する必要があります．そこで，できるだけたくさんのファイルを使える状態にしておけばようのかというとそういうわけではありません．ファイルバッファの領域はメモリ上に確保されていますので，ここで大きな数を指定するとその分ユーザーが使用できるメモリが減ってしまいます．特別に多くのファイルを同時オープンするのでなければ，あまり多く設定しても無意味です．

なお，BASICの起動時に/F：パラメータを指定しないと，2が設定されたものとみなされます．

> コンパイラの場合は，REM　$FILE命令で指定します．指定数値の範囲，省略時の値はインタプリンタと同様です．

▶**参照**　起動パラメータ→「2.3　起動パラメータの指定方法」
REM　$FILE命令→第2部の「2.2　ファイル使用のための制御命令」

# 8.2 BASIC命令によるディスクファイルの操作

BASICにはディスクファイルを扱うための命令が数多く用意されています．これらの命令を使うことによって，作成したプログラムや文書データなどをファイルとして保存し，必要に応じて呼び出して使用するといったことが自由にできるようになります．

ここではBASICの命令を使ったファイル操作の方法について解説します．

BASICのLOAD，SAVE，RUN，MERGE，CHAIN命令で〈ファイルディスクリプタ〉を指定するときに拡張子を省略すると自動的に〝.BAS〞が指定されます．

## ■ディスクの中のファイルの情報チェック

ディスクにはたくさんのファイルを記録しておくことができます．ファイルの操作をする場合にはディスクにどのようなファイルが入っているかを常に知っておく必要があります．

ディスクの中に入っているファイルの情報を調べるにはFILES命令を使います．

FILES命令

FILES 〔〈ドライブ名〉：〕〔¥〕〔〈ディレクトリ名〉¥〕…〔〈ディレクトリ名〉¥〕〔〈主ファイル名〉〔.〈拡張子〉〕〕

FILES命令を実行すると，まず指定されたドライブ名，ディレクトリ名が表示されます．つづけてそのドライブ，ディレクトリの該当するファイルについて，主ファイル名と拡張子，そのサイズ(バイト単位)および最終変更日付が表示されます．

〈ドライブ名〉，〈ディレクトリ名〉を省略した場合，それぞれカレントドライブ，カレントディレクトリが対象となります．

〈主ファル名〉，〈拡張子〉についてはワイルドカード(＊，？)を使用することができます．詳しくは「MS-DOSのマニュアル」を参照してください．〈主ファイル名〉，〈拡張子〉を省略した場合はそのディレクトリ中のすべてのファイルが対象となります．

また，FILES命令のかわりにLFILES命令を使用するとファイルの情報はプリンタに出力されます．

例) ドライブBのカレントディレクトリ下のファイルの情報を調べる．

```
FILES "B:" [↵]
B:¥
SAMPLE1  BAS   167  88/04/30   SAMPLE2  BAS   170  88/05/02
SAMPLE3  BAS   200  88/05/02   SAMPLE4  BAS   229  88/05/02
SAMPLE5  BAS   197  88/05/03
```

## ■プログラムファイルのセーブとロード

BASICを使ってプログラムを作成しても本体の電源を切る／リセットスイッチを押す／BASICを終了するとプログラムは失われてしまいます．そこで作成したプログラムをディスクに保存します．こうすれば，必要な時にディスクから取り出して使用することができます．プログラムをディスクに保存(セーブ)するための命令がSAVE命令，ディスクから取り出す(ロード)ための命令をLOAD命令といいます．また，RUN命令を使用してプログラムをロードして直ちに実行することもできます．

### ●セーブ(SAVE)

BASICのエディタ機能を使用してプログラムを作成した状態では，プログラムはメモリのBASICプログラム領域にデータとして存在しています．SAVE命令を使うと，このメモリ上のデータをディスクに書き込むことができます．

#### アスキー形式とバイナリ形式

SAVE命令ではプログラムをアスキー形式とバイナリ形式という2とおりの形式でセーブすることができます．

アスキー形式とはプログラムをLIST命令で表示したときと同じイメージでテキストとしてセーブするものです．

バイナリ形式とはプログラムをメモリにある状態のままでディスクにセーブするものです．プログラムはメモリ上ではメモリの節約をするために中間コードという圧縮された状態で存在しており，バイナリ形式によるセーブを行うとメモリの内容がそのままの状態でディスクにセーブされることになります．

#### アスキー形式の利点

アスキー形式でセーブされたプログラムファイルはテキストファイルとして扱うことができます．したがって，OPEN命令などを使ったファイルの入出力を行うことができます．また，MERGE命令を使って2つのプログラムを合成することも可能になります．

#### バイナリ形式の利点

バイナリ形式でセーブすると，通常，アスキー形式でセーブしたものに比べてプログラムファイルの大きさが小さくなりますので，ディスクを無駄なく使用することができ，また，セーブ，ロードの時間も短縮されます．

#### SAVE命令

SAVE　<ファイルディスクリプタ>　〔, {A | P}〕

Aオプションも Pオプションもつけない場合はバイナリ形式でセーブされます．

Aオプションをつけるとアスキー形式でセーブされます．Pオプションをつけるとプログラムは暗号化されたバイナリ形式でセーブされます．この形式でセーブされたプログラムファイルは，ロードしても LIST 命令や EDIT 命令を実行することはできません．

また，Pオプションを指定すると二度と解除できませんので注意してください．

---

**注意**：SAVE 命令を実行する際にすでにディスク上にあるファイルと同じファイルディスクリプタを指定すると，ディスク上のファイルは失われてしまいますので注意してください．

---

### ●ロード(LOAD)

ディスク上のプログラムファイルを BASIC のメモリ領域にロードするには LOAD 命令を使用します．この命令を実行するとメモリ上にプログラムが読み込まれ，そのプログラムが実行可能な状態になります．

LOAD 命令

```
LOAD 〈ファイルディスクリプタ〉〔, R〕
```

Rオプションをつけるとロード後ただちにプログラムを実行します．この場合，そのときオープンされていたファイルは閉じられず，オープンされたままで実行されます．

---

**注意**：LOAD 命令を実行すると，それまで BASIC のメモリ領域にあったプログラムは消滅してしまいます．LOAD 命令を実行する前にはメモリ上に重要なプログラムがないことを確認しましょう．

---

### ●ロード後実行(RUN)

普通プログラムを実行するには RUN 命令を使用しますが，RUN 命令にはディスクからプログラムをロードして，直ちに実行させるという機能もあります．

RUN 命令

```
RUN 〔〈ファイルディスクリプタ〉〕〔, R〕
```

Rオプションをつけると，そのときオープンされていたファイルは閉じられず，オープンされたままで実行されます．

## ■プログラムのマージ

メモリ上にあるプログラムとディスクの中にあるプログラムを行単位に合成して，1つのプログラムにすることを「マージ」といいます．そのための命令がMERGE命令です．プログラムの改造を行う場合などに有効です．

### MERGE命令

MERGE　〈ファイルディスクリプタ〉

マージする際，2つのプログラムに同じ行番号があるとディスク上のプログラムが優先されて，メモリ上のプログラムのその行と置き変わります．

---

注意：MERGE命令でディスクから読み込むプログラムファイルは必ずアスキー形式でセーブされたものでなければなりません．

---

## ■プログラムの連結

機能が豊富なプログラムを作ろうとするとどうしても大きなものになります．その結果，大きくなりすぎてメモリに入りきらないとか，プログラムのバグ(誤り)が発生しやすいといったことがおこりがちです．

このような問題を解決するための方法としてプログラムのモジュール化があります．プログラムを単純な機能のモジュールにわけ，それを連結(チェイン)して実行させるのです．

モジュール化は次のような手順で行います．

①プログラムをいくつかの単純な機能を持ったモジュールに分けて作成する．
②それぞれのプログラムをディスクにセーブする．
③最初のモジュールを実行させる．
④その後，自動的に必要なモジュールを次々にメモリにロードして実行する．

④の自動実行は，前に説明したRUN命令やLOAD命令のRオプションを使って行うことができますが，それよりももっと便利な命令としてCHAIN命令が用意されています．

### CHAIN命令

CHAIN　〔MERGE〕〈ファイルディスクリプタ〉〔,〈行番号〉〕〔,ALL〕
〔,DELETE〈範囲〉〕

CHAIN命令の各パラメータは次のような働きがあります．

#### 〈ファイルディスクリプタ〉

〈ファイルディスクリプタ〉で指定したプログラムファイルをメモリにロードしてから実行します．

**〈行番号〉**

プログラムをメモリにロードした後，〈行番号〉で指定した行から実行します．〈行番号〉の指定を省略すると先頭の行から実行されます．

**ALL オプション**

プログラムをメモリにロードした後，実行します．CHAIN 命令を実行する前のすべての変数の値はクリアされずそのまま新しいプログラムで使用できます．

ALL オプションを省略した場合には CHAIN 命令を実行する前のすべての変数の値はクリアされます．

なお，特定の変数のみを使えるようにするには，COMMON 命令を使用します．

▶参照　COMMON 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

**MERGE オプション**

メモリ上のプログラムとディスク上のプログラムをマージし，実行します．マージする際，2つのプログラムに同じ行番号があるとディスク上のプログラムが優先されて，メモリ上のプログラムのその行と置き変わります．

---

**注意**：MERGE オプションを使ってディスクから読み込むプログラムファイルは必ずアスキー形式でセーブされたものでなければなりません．

---

**DELETE オプション**

MERGE オプションを指定したときのみに指定することができます．これを指定すると，メモリ上のプログラムの中から DELETE オプションで指定した範囲を削除した後でマージを行い，実行します．

**例1）** ドライブ B のディスクから〝SAMPLE1.BAS″をロードし，300 行から実行する．

```
CHAIN  "B:SAMPLE1.BAS", 300
```

**例2）** メモリ上のプログラムの 200 行から 500 行を削除し，ドライブ B のディスクの〝SAMPLE1.BAS″をマージした後で実行する．CHAIN 命令を実行する前の変数の値はクリアされない．

```
CHAIN  MERGE  "B:SAMPLE1.BAS", ALL, DELETE 200-500
```

## ■機械語ファイルのセーブとロード

コンピュータが動作しているとき，メモリはデータを一時的に保存する場所となります．BASICのテキストはメモリのテキストエリアに，機械語のプログラムは機械語プログラムエリアに，また画面の情報はVRAMに保存されています．これらのデータは電源を切ったり，リセットスイッチを押せば失われてしまいます．

BASICには，このメモリのデータをそのままのイメージでディスクファイルとして保存(セーブ)したり，ディスクから逆に取り出し(ロード)たりする命令が用意されています．

これがBSAVE命令とBLOAD命令です．この命令は特に機械語プログラムのセーブ，ロードによく使われます．

▶参照　機械語プログラム→「第10章　機械語プログラム」

### ●セーブ(BSAVE)

**BSAVE命令**

BSAVE　〈ファイルディスクリプタ〉,〈開始アドレス〉,〈長さ〉

〈開始アドレス〉を先頭として〈長さ〉で表される大きさのメモリイメージを〈ファイルディスクリプタ〉で指定されたファイル名で，バイナリファイルとしてセーブします．

このとき指定する〈開始アドレス〉はセグメント内の相対アドレスです．したがって，この命令を実行する前にはDEF SEG命令でセグメントベースを設定しなければなりません．

### ●ロード(BLOAD)

**BLOAD命令**

BLOAD　〈ファイルディスクリプタ〉〔,〈ロードアドレス〉〕〔,R〕

〈ファイルディスクリプタ〉で指定されたファイルをメモリ上にロードします．

〈ロードアドレス〉の指定を省略した場合は，0番地(セグメントベースの先頭アドレス)からロードが行われます．〈ロードアドレス〉を指定すると，ロードはそのアドレスから行われます．BLOAD命令のアドレス指定はセグメント内の相対アドレスです．したがって，この命令を実行する前にはDEF SEG命令でセグメントベースを設定しなければなりません．

なお，事前にCLEAR命令を実行してロードする領域を機械語プログラム用に確保しておかなければなりません．

## ■ファイルの削除

ディスクの中にあるファイルを削除するときには，KILL 命令を使用します．

KILL 命令

KILL　〈ファイルディスクリプタ〉

KILL 命令では，MS-DOS 上のどんな種類のディスクファイルでも削除することができますが，必ずクローズされた状態でなければなりません．SET 命令でファイルやディスク全体に書き込み禁止の属性を付けた場合，該当するファイルは削除することができません．また，ライトプロテクトを施したディスク中のファイルを削除することはできません．

---

注意：KILL 命令を実行する際には，誤ってファイルを消さないように〈ファイルディスクリプタ〉の指定には十分な注意をはらってください．

---

## ■ファイル名の変更

ディスクの中にあるファイルのファイル名を別の名前に付け変えるには NAME 命令を使用します．

NAME 命令

NAME　〈旧ファイルディスクリプタ〉　AS　〈新ファイルディスクリプタ〉

MS-DOS 上のどんな種類のディスクファイルでもファイル名を変更することができますが，必ずクローズされた状態でなければなりません．また，1 つのファイルの名前を変更するのですから，〈旧ファイルディスクリプタ〉と〈新ファイルディスクリプタ〉に指定するドライブ名は同じものでなければなりません．

## ■ファイル属性の設定

SET 命令はディスク，ドライブ，ファイルバッファに対して「書き込み禁止」の属性の指定・解除を行うものです．

SET 命令

1）SET　〈ファイルディスクリプタ〉，　"P"

2）SET　#〈ファイル番号〉，　"P"

1)は指定したファイルに対してだけ属性が付けられます．また，2)は指定したファイル番号に対応するファイルバッファに対して属性が付けられます．

属性の指定には，"P"のオプションを使います．"P"を指定すると「書き込み禁止」の属性が付けられます．"p"と小文字で指定してもかまいません．

「書き込み禁止」の属性が付けられるとそのファイル，ファイルバッファに対して出力モードでのOPEN命令やランダムモード時のPUT命令などの書き込み動作が禁止されるとともに，ファイルに対するKILL命令も実行できなくなります．

属性を解除するには，空白(" ")または空の文字列をオプションとしてSET命令を実行してください．

## ■ファイルの属性チェック

ATTR$関数を使うと，SET命令で付けられたファイルの属性を調べることができます．

**ATTR$関数**

1) ATTR$(〈ファイルディスクリプタ〉)

2) ATTR$(#〈ファイル番号〉)

1)はファイルに付けられている属性を，2)はファイルバッファの属性を得ることができます．得られる値とその意味は次のとおりです．□は空白を意味します．

| | |
|---|---|
| "□□" | 属性が解除された状態です． |
| "□P" | 「書き込み禁止」の属性が付けられています． |
| "E□" | Pオプションつきで SAVE されたファイルです． |

## ■ディレクトリ管理

MS-DOSでディレクトリの作成/削除，カレントディレクトリの変更などを行うことができます．BASICでもディレクトリを管理する，MKDIR/RMDIR/CHDIR命令が用意されています．

### ●ディレクトリの作成

**MKDIR命令**

MKDIR　〔〈ドライブ名〉〕〈ディレクトリ名〉

指定されたドライブのディスクに〈ディレクトリ名〉で指定されるディレクトリを作成します．〈ドライブ名〉が省略された場合，カレントドライブが処理対象となります．

例1)　カレントドライブ，カレントディレクトリの下に"BASIC1"というディレクトリを作成する．

```
MKDIR "BASIC1"
```

例2） ドライブB，ルートディレクトリの下に〝BASIC2〞というディレクトリを作成する．

MKDIR "B : ¥BASIC2"

例3） 例2で作成した〝BASIC2〞というディレクトリの下に〝BASIC3〞というディレクトリを作成する．

MKDIR "B : ¥BASIC2¥BASIC3"

**●ディレクトリの削除**

**RMDIR 命令**

RMDIR 〔〈ドライブ名〉〕〈ディレクトリ名〉

指定されたドライブのディスクから，〈ディレクトリ名〉で指定されるディレクトリを削除します．ドライブ名が省略された場合，カレントドライブが処理対象となります．

また，カレントディレクトリや空でないディレクトリ（ディレクトリの下にファイルやディレクトリが存在する）を削除することはできません．

例） カレントドライブのカレントディレクトリの下の〝BASIC1〞という空のディレクトリを削除する．

RMDIR "BASIC1"

**●カレントディレクトリの変更**

**CHDIR 命令**

CHDIR 〔〈ドライブ名〉〕〈ディレクトリ名〉

カレントディレクトリを〈ディレクトリ名〉で指定されるディレクトリに変更します．ドライブ名が省略された場合，カレントドライブが処理対象となります．

例） ドライブBのカレントディレクトリをルートディレクトリの下の〝BASIC2〞に変更する．

CHDIR "B : ¥BASIC2"

## 8.3 データファイルの扱い方

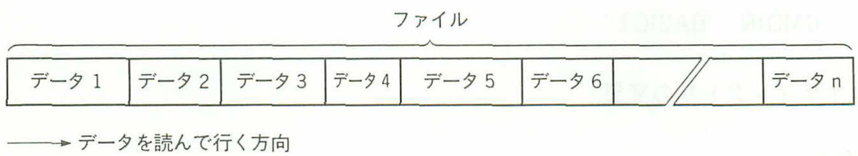
データファイルにはシーケンシャルファイルとランダムファイルの2種類があります．

### ■シーケンシャルファイルとランダムファイル

シーケンシャルファイルは連続して書かれたデータの集まりで次のような特徴があります．

- 読み書きするデータの長さは自由に変更できる．
- ファイル操作が簡単である．
- データの読み書きは必ず先頭からしかできない．データを捜すには先頭から最後まで捜さなければならないので時間がかかる．
- ファイル内の一部のデータだけを変更することができない．変更するときはファイル全部を更新するほかない．
- データの追加はファイルの終わりに書き込むことしかできない．

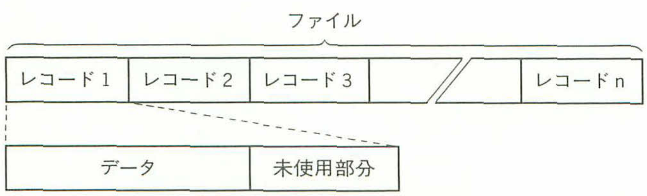
**シーケンシャルファイルの構造**




いっぽう，ランダムファイルはある一定の大きさのカードにデータを書き込んだものを並べたようなもので，次のような特徴があります．

- ファイル内のデータを自由に何番目のものでも自由に読み書きすることができので，必要なデータだけを即座に取り出すことができる．
- ファイル内の特定のデータだけを変更することができる．
- シーケンシャルファイルに比べて，プログラムがやや複雑になる．
- 読み書きするデータの大きさは固定されている．

**ランダムファイルの構造**




データはレコードを単位にして読み書きされる．

シーケンシャルファイルもランダムファイルもそれぞれ長所と短所がありますので，その違いを十分理解して処理の内容にあったファイルを作成する必要があります．

## ■シーケンシャルファイルの操作

まず比較的操作が簡単なシーケンシャルファイルから解説します．

シーケンシャルファイルの操作手順には基本的な形式があります．それは次のとおりです．

①ファイルをオープンする．

②データを読み書きする．

③ファイルをクローズする．

各段階において使われる命令を簡単に説明します．

### ●ファイルをオープンする

**OPEN 命令**

OPEN 〈ファイルディスクリプタ〉〔FOR〈入出力モード〉〕 AS 〔#〕〈ファイル番号〉

OPEN 命令の機能として次の 2 つがあります．

**ファイルにファイルバッファを割り当てる**

〈ファイルディスクリプタ〉で指定されたファイルに〈ファイル番号〉で表される専用ファイルバッファを割り当てます．以後入出力操作を行うときにはこの〈ファイル番号〉を指定します．

**ファイルのアクセスモードを設定する**

〈入出力モード〉はファイルのアクセスの仕方を指定するものです．シーケンシャルファイルの場合は次の 3 種類があります．

INPUT ………………入力モード．既存のファイルよりデータを読み込むことを示す．

OUTPUT ……………出力モード．新規にファイルを作り，書き込みを行うことを示す．

APPEND ……………追加モード．既存のファイルに追加書き込みを行うことを示す．

OPEN 命令を実行することによってファイルが入出力可能となります．

---

**注意**：入力モード，追加モードでオープンした場合はファイルがないとエラーになります．また，既にあるファイルを出力モードでオープンするとそのファイルは失われてしまいます．

---

### ●データを読み書きする

**PRINT #，PRINT # USING，WRITE #命令**

PRINT #〈ファイル番号〉，〔〈式〉〕〔｛，／；｝〈式〉…〕〔｛，／；｝〕

PRINT #〈ファイル番号〉，USING〈書式制御文字列〉；〈式〉〔｛，／；｝〈式〉…〕〔｛，／；｝〕

WRITE #〈ファイル番号〉，〈式〉〔｛，／；｝〈式〉…〕〔｛，／；｝〕

これらはファイルバッファへの書き出しを行う命令です．これらの命令を使って〈ファイル番号〉で指定されるファイルバッファへ〈式〉の値を書き込みます．ファイルバッファがいっぱいになったときにファイルへの書き込みが行われます．

基本的な機能は3つとも同じです．それぞれの命令の違いについては「N$_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」を参考にしてください．なお，これらの命令はファイルが出力モードあるいは追加モードでオープンされていなければ使用できません．

**INPUT #，LINE INPUT #命令，および INPUT $関数**

INPUT #〈ファイル番号〉，〈変数名〉〔，〈変数名〉…〕
LINE INPUT #〈ファイル番号〉，〈文字型変数名〉
INPUT $(〈文字数〉〔，〔#〕〈ファイル番号〉〕)

ファイルバッファからの読み込みを行う命令・関数です．

これらの命令・関数を使って，ファイルバッファからデータを読み込むことができます．

基本的な機能は3つとも同じです．それぞれの命令の違いについては「N$_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」を参考にしてください．なお，これらの命令はファイルが入力モードでオープンされていなければ使用できません．

そのほかにファイルの状態を調べるための命令・関数が用意されています．

**EOF関数**

EOF(〈ファイル番号〉)

〈ファイル番号〉で指定されたファイルのデータを最後まで読み込んだら真(−1)，そうでなくまだ残っているなら偽(0)を与えます．ファイルの読み込みが終わっているかどうかの判定に使用します．なお，この関数はファイルが入力モードでオープンされていなければ使用できません．

### LOC 関数

LOC(<ファイル番号>)

<ファイル番号>で指定されたファイルをオープンしてから現在までにファイルの入出力をした回数が得られます.

### LOF 関数

LOF(<ファイル番号>)

<ファイル番号>で指定されたファイルの大きさをバイト数で得ます.

### ●ファイルをクローズする

### CLOSE 命令

CLOSE 〔[#]<ファイル番号>[,〔#]<ファイル番号>]…〕

CLOSE 命令には次の 2 つの機能があります.

**ファイルバッファを解放する**

ファイルに割り当てられていたファイルバッファを解放し，別のファイルで使用できるようにします.

**最後にファイルバッファに残ったデータをファイルに出力する**(出力モードおよび追加モードのとき)

データの出力命令は実際にはファイルバッファへデータ出力を行い，ファイルバッファがいっぱいになったらファイルに書き出すようにしています．したがって，ファイルバッファにデータが残っている可能性があります．このデータをファイルに出力します.

## ■シーケンシャルファイルの例

シーケンシャルファイルの例として住所録を取り上げています.

### ●住所録の作成

キーボードから名前と住所と電話番号を入力してファイルを作成します.

```
100 OPEN "ADDRES.DAT" FOR OUTPUT AS #1 ―――― "ADDRES.DAT" を出力モードでオープンする.
110 INPUT "名前";A$ ――――――――――――――――――― A$に名前を入力する.
120 IF A$="" THEN CLOSE:END ―――――――――――― 110行で⏎キーのみを押したら終了する.
130 INPUT "住所";B$ ――――――――――――――――――― B$に住所を入力する.
140 INPUT "電話番号";C$ ――――――――――――――――― C$に電話番号を入力する.
150 PRINT #1,A$;",";B$;",";C$ ――――――――― データをファイルに書き出す.
160 PRINT ―――――――――――――― 改行する.
170 GOTO 110 ――――――――――――― 110行へもどる.
```

この場合，漢字でデータを入力するとファイルには次のような形式でデータが出力されます.

| 文字および制御コード | 鈴 | | 木 | | 太 | | 郎 | | , | 東 | | 京 | | 都 | | , | 0 | 3 | - | 1 | 1 | 7 | - | 4 | 3 | 2 | 1 | CR | LF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16進表記 | 97 | E9 | 96 | D8 | 91 | BE | 98 | 59 | 2C | 93 | 8C | 8B | 9E | 93 | 73 | 2C | 30 | 33 | 2D | 31 | 31 | 37 | 2D | 34 | 33 | 32 | 31 | 0D | 0A |

## ●住所録の画面表示

住所録の中身をすべて画面表示します.

```
100 OPEN "ADDRES.DAT" FOR INPUT AS #1 ――――― "ADDRES.DAT" を入力モードでオープンする.
110 IF EOF(1) THEN CLOSE:END ――――ファイルの中身がなくなったら終了する.
120 INPUT #1,A$,B$,C$ ―――――――――ファイルからデータを読み込む.
130 PRINT "名前：";A$ ―――――――――名前を表示する.
140 PRINT "  住所：";B$ ――――――――住所を表示する.
150 PRINT "  電話：";C$ ――――――――電話番号を表示する.
160 PRINT ―――――――――改行する.
170 GOTO 110 ――――――――110行へもどる.
```

## ●住所録の検索

住所録の中から特定の文字，数字を含む人のデータを表示します.

```
100 INPUT "捜す文字は";F$:PRINT ――――――検索する文字を入力する.
110 OPEN "ADDRES.DAT" FOR INPUT AS #1 ―――― "ADDRES.DAT" を入力モードでオープンする.
120 IF EOF(1) THEN CLOSE:END ―――――――――ファイルの中身がなくなったら終了する.
130 INPUT #1,A$,B$,C$ ――――――――――――――データをファイルから読み込む.
140 IF KINSTR(A$+B$+C$,F$)=0 THEN 120 ――――データの中に検索する文字がなかったら120行
150 PRINT "名前：";A$ ――――――――――名前を表示する.                     へもどる.
160 PRINT "  住所：";B$ ―――――――――住所を表示する.
170 PRINT "  電話：";C$ ―――――――――電話番号を表示する.
180 PRINT ――――――――――改行する.
190 GOTO 120 ―――――――――120行へもどる.
```

## ●住所録の追加

住所録にデータを追加します.

```
100 OPEN "ADDRES.DAT" FOR APPEND AS #1 ――――― "ADDRES.DAT" を追加モードでオープンする.
110 INPUT "名前";A$ ――――――――――A$に名前を入力する.
120 IF A$="" THEN CLOSE:END ――――110行で⏎キーのみを押したら終了する.
130 INPUT "住所";B$ ――――――――――B$に住所を入力する.
140 INPUT "電話番号";C$ ――――――――C$に電話番号を入力する.
150 PRINT #1,A$;",";B$;",";C$ ―――データをファイルに書き出す.
160 PRINT ――――――――――改行する.
170 GOTO 110 ―――――――――110行へもどる.
```

## ■ランダムファイルの操作

ランダムファイルを操作する場合は，あらかじめバッファサイズを指定することができます．バッファサイズは次のように指定します．

- ・インタプリンタの場合は起動時に指定します．詳しくは「2.3　起動パラメータの指定」を参照してください．

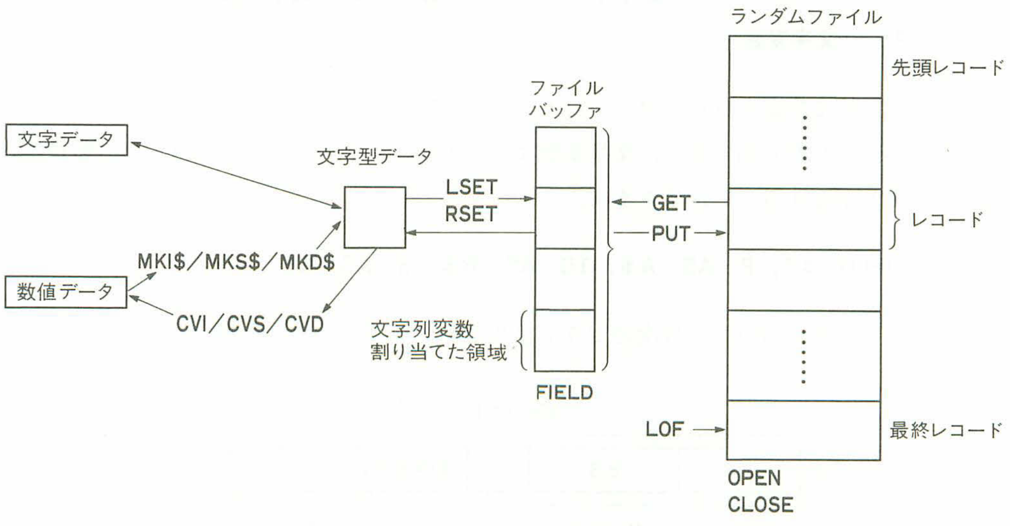
- ・コンパイラの場合はREM　$FILE命令で指定します．詳しくは第2部の「2.2 ファイル使用のための制御命令」を参照してください．

ランダムファイルを操作する手順は次のようになります．

①ファイルをオープンする．
②ファイルバッファに文字変数(フィールド変数)を割り当てる．
③バッファにデータをセットし，ファイルへ書き出す．
④データをバッファに読み込み，バッファに割り当てられた文字変数からデータを取り出す．
⑤ファイルをクローズする．

なお，③の操作において数値データを扱う場合は，前もって数値データを文字型データに変換(MKI$，MKS$，MKD$関数)してからバッファにセットします．またこのような変換を行ったデータに対しては，④の操作においてデータを取り出したあとで文字型データを数値データにもどします(CVI，CVS，CVD関数)．

**ランダムファイルのデータの受け渡し**




各段階において使われる命令・関数を簡単に解説します．

### ●ファイルをオープンする

**OPEN 命令**

　　OPEN　〈ファイルディスクリプタ〉　AS　〔#〕〈ファイル番号〉

　ランダムファイルではシーケンシャルファイルのようなモードの指定はありません．〈ファイルディスクリプタ〉で指定したファイルの入出力のために〈ファイル番号〉のファイルバッファが割り当てられ，以後，この〈ファイル番号〉を指定してデータの入力・出力の両方ができるようになります．

### ●フィールド変数を割り当てる

　シーケンシャルファイルでは読み書きするデータの長さが自由でしたが，ランダムファイルではデータの大きさはファイルバッファのサイズに固定されています．これを1レコードといいます．

　1レコードの大きさはファイルバッファと同じですから，一度にファイルバッファの大きさの分だけデータの受け渡しができるということになります．

　また，ファイルバッファの領域をいくつかの領域に分けることにより，複数のデータの受け渡しを一度に行うことができます．

　FIELD命令はファイルバッファに複数のフィールド変数の領域を割り当てます．その結果この変数を使って，プログラムとファイルバッファ間のデータの入出力を行うことができるようになります．

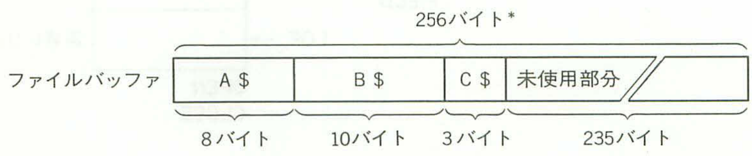
**FIELD 命令**

　　FIELD　〔#〕〈ファイル番号〉，〈フィールド幅〉　AS　〈文字変数〉〔，〈フィールド幅〉
　　AS　〈文字変数〉…〕

　〈ファイル番号〉で指定されたファイルバッファに〈フィールド幅〉バイトの大きさの文字変数(フィールド変数)を割り当てます．文字変数はファイルバッファの大きさを超えない範囲であればいくつでも指定することができます．

　　FIELD　#1，8　AS　A$，10　AS　B$，3　AS　C$

を実行するとファイルバッファは次のようになります．




*：ファイルバッファのサイズが256バイトの場合．

FIELD 命令によるフィールド変数の割り当ては1つのファイルバッファに対して何回でも行うことができます．これらの定義はすべてが有効になります．

**●ファイルバッファにデータをセットしてファイルへ書き出す**

ファイルに対する出力を行うには，まず，ファイルバッファにデータをセットします．

ファイルバッファへのデータのセットは，LSET/RSET 命令を使って，ファイルバッファに割り当てたフィールド変数に文字列データを代入するという方法で行います．

**LSET/RSET 命令**

LSET　〈文字変数〉＝〈文字列〉

RSET　〈文字変数〉＝〈文字列〉

〈文字列〉の文字数が〈文字変数〉(FIELD 命令で割り当てられたフィールド変数)の長さより短い場合は，LSET では左詰め，RSET では右詰めとなります．

**PUT 命令**

PUT　〔#〕〈ファイル番号〉〔，〈数式〉〕

〈ファイル番号〉で指定されたファイルバッファの内容を対応するファイルに書き出します．ここで初めてディスクファイルへの書き出しが行われます．〈数式〉にはファイル中のレコード番号を指定します．レコード番号の最小値は1，最大値は65535です．PUT 命令を実行するとディスク上のある1レコード分の領域にデータが書き出され，各々にレコード番号がふられることになります．この番号はデータを GET 命令で参照したり，PUT 命令で変更したりする場合の目印となるものです．レコード番号に大きな数を指定するとその数の分だけのレコードが設定されます．たとえば最初に300と指定して PUT 命令を実行すると，ディスクに1～300までのレコード番号が割り振られ，その大きさの領域がランダムファイルとして確保されます．この場合1～299までのレコードにはデータは書き込まれていませんので，1～299までのレコードを GET 命令で参照した場合のデータは意味のないものとなります．

レコード番号の指定を省略した場合は，直前に行われた PUT 命令あるいは GET 命令で使われたレコードの次のレコードに書き込みが行われます．

---

**注意**：レコード番号の最大値は65535ですが，ディスクの空領域が限られていますので，実際にレコード番号として指定できる値はより小さくなります．レコード番号として指定できる限界値を超えて指定を行うとエラーが出ます．

---

●ファイルからデータをバッファに読み込み，バッファからデータを取り出す

GET 命令

　　　GET　〔#〕<ファイル番号>〔,<数値>〕

ファイルに書き込まれているデータを〈ファイル番号〉で指定されたファイルバッファに読み込みます．〈数値〉には読み込みたいレコード番号を指定します．レコード番号の指定を省略した場合は直前に行われたPUT命令あるいはGET命令で使われたレコードの次のレコードから読み込みが行われます．

バッファに割り当てられたフィールド変数にデータが代入されます．

●ファイルをクローズする

ランダムファイルのクローズもCLOSE命令を使用します．

CLOSE 命令

　　　CLOSE　〔〔#〕<ファイル番号>〔,〔#〕<ファイル番号>〕…〕

CLOSE命令を実行するとファイルに割り当てられていたファイルバッファが解放され，別のファイルで使用できるようになります．

●その他の命令・関数

LOF 関数

　　　LOF(<ファイル番号>)

〈ファイル番号〉で指定されるファイルの最大のレコード番号，つまりファイルの大きさをレコード数で得ることができる関数です．LOC関数といっしょに使用してファイルが最後まで読み込まれているかの判定を行うことができます．

LOC 関数

　　　LOC(<ファイル番号>)

〈ファイル番号〉で指定されるファイルにおいて，そのファイルを最後に読み書きしたレコード番号を得ることができます．前の例のLOF関数といっしょに使用してファイルが最後まで読み込まれたかどうかの判定をすることができます．

例）　最後に読み込まれたレコード番号が最後のレコード番号と同じであればファイルをクローズする．

```
IF  LOC(1)=LOF(1)  THEN  CLOSE  #1:END
```

**MKI＄/MKS＄/MKD＄関数**

　　MKI＄(〈整数値〉)……………………整数値を2文字(2バイト)の文字列に変換
　　MKS＄(〈単精度実数値〉)　………単精度実数値を4文字(4バイト)の文字列に変換
　　MKD＄(〈倍精度実数値〉)　………倍精度実数値を8文字(8バイト)の文字列に変換

ランダムファイルへの書き込みは，文字列で行わなければなりません．したがって数値を書き込みたいときには文字列に変換してからLSET/RSET命令でフィールド変数に代入するようにします．数値の種類によって文字列の長さが変わります．

**CVI/CVS/CVD関数**

　　CVI(〈2文字の文字列〉)　…………2文字の文字列を整数値に変換
　　CVS(〈4文字の文字列〉)…………4文字の文字列を単精度実数値に変換
　　CVD(〈8文字の文字列〉)…………8文字の文字列を倍精度実数値に変換

ランダムファイルの書き込みの際にMKI＄/MKS＄/MKD＄関数で変換された文字列を，読み込みの際に元の数値にもどすために使用します．

## ■ランダムファイルの例

ランダムファイルの例として名簿を取り上げます．

### ●名簿の作成

キーボードから名前，出身地，年齢，登録番号を入力してランダムファイルを作成します．この例では登録番号がレコード番号と同じ数値になっています．

```
100 OPEN "MEIBO.DAT" AS #1 ―――――――――― "MEIBO.DAT"をオープンする.
110 FIELD #1,30 AS AA$,16 AS BB$,2 AS CC$, 4 AS NO$ ――ファイルバッファにフィールド
120 *NEXTP                                                  変数を割り当てる.
130 PRINT ―――――――――――――――――――改行する.
140 INPUT "名前    ";A$ ――――――――――名前を入力する.
150 IF A$="" THEN CLOSE:END ――――――140行で[⏎]キーだけが押されたら終了する.
160 INPUT "出身地  ";B$ ――――――――――出身地を入力する.
170 INPUT "年齢    ";C% ――――――――――年齢を入力する.
180 INPUT "登録番号";N ―――――――――――登録番号を入力する.
190 LSET AA$=A$ ―――――――――――――――データをバッファにセットする.
200 LSET BB$=B$
210 LSET CC$=MKI$(C%)
220 LSET NO$=MKS$(N)
230 PUT #1,N ――――――――――データをファイルに書き出す.
240 GOTO *NEXTP ―――――――――120行へもどる.
```

### ●名簿の表示

登録番号を指定してデータを画面に表示させます．

```
100 OPEN "MEIBO.DAT" AS #1 ――――――― "MEIBO.DAT" をオープンする.
110 FIELD #1,30 AS AA$,16 AS BB$,2 AS CC$, 4 AS NO$ ――ファイルバッファにフィールド
120 *NEXTP                                                 変数を割り当てる.
130 PRINT ―――――――――――――――――――――改行する.
140 INPUT "登録番号は";N$ ――――――――――登録番号を入力する.
150 IF N$="" THEN CLOSE:END ―――――――――140行で⏎キーのみが押された場合には終了する.
160 IF VAL(N$)>LOF(1) THEN *NEXTP ――――――登録番号が登録されているデータの最大レコード番号
170 GET #1,VAL(N$) ――――――――データをファイルから          より大きい場合は120行へもどる
180 PRINT                          ファイルバッファへ読み込む.
190 PRINT "  登録番号 ：";CVS(NO$) ―――――データを表示する.
200 PRINT "  名前     ：";AA$
210 PRINT "  出身地   ：";BB$
220 PRINT "  年齢     ：";CVI(CC$)
230 GOTO *NEXTP ――――――――――120行へもどる.
```

# 8.4　ファイルの共用制御

MS-DOS(Ver 3.1以降)上でMS-NETWORKSが使用されている環境では，BASICでファイルの共用制御を行うことができます．

この場合，OPEN命令で〈共用モード〉の指定が可能となります．

**OPEN命令**

OPEN　〈ファイルディスクリプタ〉〔FOR〈入出力モード〉〕〔〈共用モード〉〕AS〔#〕〈ファイル番号〉

▶**参照**　MS-NETWORKS→「MS-NETWORKSのマニュアル」

〈共用モード〉には次の5種類の指定方法があります．

・**DENY READ**

このモードを指定されたファイルがオープンされている間，他プロセス／自プロセスにかかわらず，データの読み出しを伴うこのファイルのオープンは禁止されます．

・**DENY WRITE**

このモードを指定されたファイルがオープンされている間，他プロセス／自プロセスにかかわらず，データの書き込みを伴うこのファイルのオープンは禁止されます．このモードでオープンされて以降，INPUTモードのみ同時オープンが可能です．OUTPUT／APPEND／入出力モード指定なしによるオープンは拒否されます．

・**DENY ALL**

このモードを指定されたファイルがオープンされている間，他プロセス／自プロセスおよび入出力モードにかかわらず，このファイルの同時オープンは拒否されます．

・**DENY NONE**

入出力とも可能です．

・**省略時**

このファイルがオープンされている間，入出力モードにかかわらず，他プロセスによるこのファイルの同時オープンは拒否されます．このモードはファイルの共用制御を行いません．通常のOPEN命令と同じです．

次に〈共用モード〉と〈入出力モード〉の関係を示します．

| 他プロセス | | INPUTモードによる先行オープン（自プロセス） | | | | | OUTPUTモードによる先行オープン（自プロセス） | | | | | APPEND/モード指定なしモードによる先行オープン(自プロセス) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | DENY READ | DENY WRITE | DENY ALL | DENY NONE | 省略 | DENY READ | DENY WRITE | DENY ALL | DENY NONE | 省略 | DENY READ | DENY WRITE | DENY ALL | DENY NONE | 省略 |
| INPUT | DENY READ | × | × | × | × | × | × | ○ | × | ○ | × | × | × | × | × | × |
| | DENY WRITE | × | ○ | × | ○ | × | × | × | × | × | × | × | × | × | × | × |
| | DENY ALL | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| | DENY NONE | × | ○ | × | ○ | × | × | ○ | × | ○ | × | × | ○ | × | ○ | × |
| | 省略 | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| OUTPUT | DENY READ | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| | DENY WRITE | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| | DENY ALL | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| | DENY NONE | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| | 省略 | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| APPEND | DENY READ | × | × | × | × | × | × | × | × | ○ | × | × | × | × | × | × |
| | DENY WRITE | × | × | × | ○ | × | × | × | × | × | × | × | × | × | × | × |
| | DENY ALL | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| | DENY NONE | × | × | × | ○ | × | × | × | × | ○ | × | × | × | × | ○ | × |
| | 省略 | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| 省略 | DENY READ | × | × | × | × | × | × | × | × | ○ | × | × | × | × | × | × |
| | DENY WRITE | × | × | × | ○ | × | × | × | × | × | × | × | × | × | × | × |
| | DENY ALL | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| | DENY NONE | × | × | × | ○ | × | × | × | × | ○ | × | × | × | × | ○ | × |
| | 省略 | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |

(○：他プロセスオープン可　　×：他プロセスオープン不可)

## 8.5 デバイスファイルの扱い方

キーボード，スクリーン(画面)，プリンタ，RS-232C回線はデバイスファイルとして扱うことができます．各々をファイルとして見た場合，キーボードファイル，スクリーンファイル，プリンタファイル，RS-232C回線ファイルと呼びます．

デバイスファイルは，ディスクファイルと同様にBASICの命令を使って入出力を行うことができますが，キーボードファイルは入力専用，プリンタファイルとスクリーンファイルは出力専用です．また，RS-232C回線ファイルは入出力とも可能ですが，ファイルをオープンする際に通信のためのプロトコル(手順)を指定する必要があります．

### ■ RS-232C回線ファイルの使い方

RS-232C回線とは内蔵されたRS-232Cインタフェースに外部機器を接続して外部とのデータ通信を行うためのものです．

RS-232Cを使用したデータ通信の代表的なものとしては，モデムと電話回線を介して他のコンピュータとの通信を行うパソコン通信がありますが，RS-232Cインタフェースを備えたプロッタ，イメージスキャナーなどの機器に直接接続してデータ通信を行うこともできます．

BASICではこのRS-232C回線をファイルとして扱い，ファイルの入出力命令を使ってデータ通信を行うことができます．

#### ● RS-232C回線ファイルのオープンと通信形式の設定

RS-232C回線を使用してデータ通信を行うには，まず，RS-232C回線ファイルをオープンします．

ファイルのオープンにはディスクファイルと同様にOPEN命令を使用しますが，RS-232C回線ファイルの場合には，データ通信のため必要なパラメータを指定する必要があります．

OPEN命令

OPEN "COM〔<回線番号>〕:〔<パリティチェック>〔<データビット長>〔<ストップビット長>〔<Xパラメータ>〔<Sパラメータ>〕〕〕〕〕" AS 〔#〕<ファイル番号>

各パラメータ指定値とその意味との関係を示します．

| パラメータ | 指定値 | 意　味 |
|---|---|---|
| 回線番号 | 1または省略<br>2<br>3 | RS-232C 第1回線<br>RS-232C 第2回線(RS-232C 拡張インタフェースボード必要)<br>RS-232C 第3回線(RS-232C 拡張インタフェースボード必要) |
| パリティチェック | E<br>O<br>N | 偶数<br>奇数<br>使わない |
| データビット長 | 7<br>8 | 7ビット<br>8ビット |
| ストップビット長 | 1<br>2<br>3 | 1ビット<br>1.5ビット<br>2ビット |
| Xパラメータ | X<br>N | 有効(受信データのフロー制御を行う)<br>無効(受信データのフロー制御を行わない) |
| Sパラメータ | S<br>N | 有効(データビット長7でカナを扱う)<br>無効(データビット長7でカナを扱わない) |

**注意**：回線番号に1以外を指定する場合は，RS-232C 拡張インタフェースボードが必要です.

これらの値は接続する機器の設定にあわせて，SWITCH コマンドやセットアップユーティリティで設定します．さらにボーレート(通信速度)，同期クロック，DEL コード受信処理についての設定が必要です.

**ボーレートの設定**

ボーレートとは，1秒当たり何ビットのデータを送受信するかという通信速度です．この設定はSWITCH コマンドまたはユーティリティ〝N88SETUP〞で行います.

▶**参照**　SWITCH コマンド→「MS-DOS のマニュアル」
N88SETUP →「12.2　セットアップユーティリティ」

**同期クロックの設定**

同期クロックとは，RS-232C インタフェースで接続した機器とコンピュータ本体との間でデータ通信を行う際のタイミングのとり方を，コンピュータ本体のタイミング(内部同期)で行うか，接続した機器のタイミング(外部同期)に合わせるかを決めるものです．普通は内部同期にします．ディップスイッチを使って設定を行います.

▶**参照**　ディップスイッチ→本体添付の「ガイドブック」

外部同期を使用するためには，RS-232C をシンクロナスモードで使用する必要があります. BASIC では外部同期の通信機能を提供していませんので，機械語の IN または OUT 命令を使用して，RS-232C インタフェースの初期化を行い，また新しい手順，コマンドなども作成しなければなりません.

**DEL コードの処理**

DEL コード(7FH または FFH)を受信した場合に，これを DEL コードとして処理するか NULL コードとして処理するかを決めることができます．この設定は，セットアップユーティリティ〝N88SETUP〟で行います．

▶**参照**　N88SETUP →「12.2　セットアップユーティリティ」

RS-232C 回線ファイルがオープンされると，以後 RS-232C 回線への入出力は〈ファイル番号〉で指定されたファイルバッファを使用して行われます．また，次に説明する命令や関数を使用して入出力操作ができるようになります．

データ通信を終了するには，CLOSE 命令を実行します．

## ● RS-232C 回線ファイルの入出力に使用できる BASIC 命令と関数

RS-232C 回線ファイルを使ったデータの入出力には次のような命令・関数を使用することができます．ここでは，各命令の機能の概略を示します．詳しくは「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」を参照してください．

| | |
|---|---|
| BLOAD 命令 | 機械語プログラムを受信する |
| BSAVE 命令 | 機械語プログラムを送信する |
| CLOSE 命令 | ファイルをクローズする |
| EOF 関数 | RS-232C 回線ファイルバッファが空になると値を真(−1)とする |
| INPUT #命令 | RS-232C 回線ファイルバッファからデータを入力(受信)する．INPUT #命令のデータの区切りは，読み込む変数が文字型の場合はコンマ(，)，改行コード($C_R$)，ダブルクォーテーション(")のいずれか，非文字型ではコンマ(，)，改行コード($C_R$)，スペース(空白文字)のいずれか．これらの区切りを受信するまでデータの入力を待つ |
| INPUT $ 関数 | RS-232C 回線ファイルバッファから指定された文字数だけ入力(受信)する |
| LINE　INPUT #命令 | RS-232C 回線ファイルバッファからデータを入力(受信)する．LINE INPUT #命令のデータの区切りは改行コード($C_R$)だけ．($C_R$)以外はすべてデータとして入力する．($C_R$)を受信するまでデータの入力を待つ |
| LOAD 命令 | プログラムを受信する．プログラムの受信が終了しても LOAD 命令は自動的には終了しない．終了させるには STOP キーを押す |
| LOC 関数 | RS-232C 回線ファイルバッファ中に受信されているデータのバイト数を得る |
| LOF 関数 | RS-232C 回線ファイルバッファの残りバイト数を得る |
| OPEN 命令 | 通信の形式，ファイル番号を定義し，RS-232C 回線ファイルバッファをオープンする |
| PRINT #命令 | データを出力(送信)する |
| PRINT #　USING 命令 | データを編集して出力(送信)する |
| SAVE 命令 | メモリ上のプログラムの出力(送信)を行う |
| WIDTH 命令 | 出力するデータの１行の長さを決める |
| WRITE #命令 | PRINT #命令と同様データを出力する．ただし区切り記号としてコンマ(，)を必ず出力し，不要な空白の出力は行わない |

### ● RS-232C 回線ファイルの割り込み処理

RS-232C 回線ファイルを通じてデータ通信を行うプログラム中で，INPUT #，LINE　INPUT #，INPUT $ 命令が実行されると，外部からのデータの入力が完了する（区切り記号を受け取る）までプログラムは停止します．相手からのデータが来ないうちは，こちらからデータを送ったり，他の作業を行ったりすることがまったくできなくなります．

この問題を解決するために用意されている命令が ON　COM　GOSUB 命令と COM　ON/OFF/STOP 命令です．この命令を使用すると，データが受信されるとあらかじめ指定した行から始まるサブルーチンへジャンプすることができますから，データが受信されない間は必要な作業を行い，データを受信した場合だけ入力処理のサブルーチンへジャンプして INPUT $ 命令を実行するといったことが可能になります．

---

**注意**：INPUT #，LINE　INPUT #は割り込みサブルーチン内では使用しないでください．データを正しく受信できない場合があります．

---

### ON　COM　GOSUB　命令

ON　COM〔(〈回線番号〉)〕　GOSUB　〈行番号〉

〈回線番号〉に指定された RS-232C 回線ファイルでデータを受信すると，〈行番号〉で始まるサブルーチンにジャンプします．

### COM　ON/OFF/STOP 命令

COM〔(〈回線番号〉)〕 ｛ON｜OFF｜STOP｝

〈回線番号〉で指定された RS-232C 回線ファイルからの割り込みを許可／禁止／停止します．

```
10 OPEN "COM:E72XS" AS #1
20 ON COM GOSUB *REC
30 COM ON
40 OPEN "TEST" FOR INPUT AS #2
50 IF EOF(2) THEN *WAIT.T
60 LINE INPUT #2,A$
70 PRINT A$;
80 GOTO 50
90 *WAIT.T
100 CLOSE #2
110 GOTO 110
1000 *REC
1010 IF LOC(#1)=0 THEN RETURN
1020 B$=INPUT$(LOC(1),#1)
1030 PRINT B$;
1040 GOTO *REC
```

データの受信がない間は，〝TEST〟というディスクファイルの内容を画面に表示します．データの受信があるとそれまでの作業を中止して*REC 行にジャンプし，送られてきたデータを画面に表示します．データの受信が終わるとメインルーチンにもどります．ディスクファイル〝TEST〟の内容の画面表示が先に終わってしまった場合，メインルーチンは＊WAIT.T にジャンプしてディスクファイルをクローズし，110 行で RS-232C 回線からのデータの受信を待ちます．

## ■ RS-232C 回線ファイル以外のデバイスファイルの使い方

### ●キーボードファイル(KYBD：)

キーボードファイルは入力モードでのみ使えます．

例） キーボードファイルを入力モードでオープンし，タイプした文字を A$ に代入する．

```
OPEN "KYBD:" FOR INPUT AS #1
INPUT #1, A$
```

### ●スクリーンファイル(SCRN：)

スクリーンファイルは出力モードでのみ使えます．

例） スクリーンファイルを出力モードでオープンし，〝ABC〟という文字を出力する．

```
OPEN "SCRN:" FOR OUTPUT AS #1
WRITE #1, "ABC"
```

---

注意：スクリーンファイルに対しては日本語表示は行えません．

---

### ●プリンタファイル(LPT：)

プリンタファイルは出力モードでのみ使えます．

例） プリンタファイルを出力モードでオープンし，〝XYZ〟という文字を出力する．

```
OPEN "LPT:" FOR OUTPUT AS #1
PRINT #1, "XYZ"
```

### ●デバイスファイルの有効利用

入出力装置をファイルとして扱うことの利点は，異なった装置に対しても，同じ命令で入出力が可能になるということです．

たとえば，PRINT #はキーボードファイル以外のすべてのファイルに対して出力を行うことができます．また，INPUT #はスクリーンファイルとプリンタファイル以外のファイルからの入力を行うことができます．この場合，入出力先の指定さえ変更すればよいことになります．

たとえば，画面に文字を表示する場合に使用する一般的な命令であるPRINT命令を数多く使ったプログラムを作ったとします．このプログラムを画面表示するかわりにプリンタに出力するように改造したい場合には，プログラム中のすべてのPRINT命令をLPRINT命令に変更しなければなりません．

それに対して，画面をスクリーンファイルとして扱い，OPEN命令で〝SCRN：″をオープンしてPRINT #命令で出力するというプログラムを作っておけば，OPEN命令の〝SCRN：″を〝LPT：″に変更するだけでプリンタに印字させることができます．さらに，出力先をディスクファイルに変えればディスクに書き込むこともできますし，〝COM：″を指定すればRS-232C回線に出力することができます．

この機能は特にプログラムのデバッグの際に有効です．次にその一例をあげます．

・RS-232C回線ファイルから送られてきたテキストファイルを入力してディスクに出力するプログラムを作ったが，書き込みがうまくいかない場合．

出力先を〝SCRN：″に変更して，どんな文字がRS-232C回線から送られてきているか調べる．

入力先を〝KYBD：″に変えてみて，キーボードから入力したデータが，うまくディスクに書き込まれるかどうか調べる．

---

**注意**：ファイルの入出力に使用可能なBASICの命令はファイルの種類によって異なりますので注意してください．

---

# 第9章

# マウス

## 9.1 マウスの概要

BASICで使用できる入力装置として，キーボードの他にマウスがあります.

マウスはディスプレイ上のポインティングデバイス(指示装置)として，強力な働きをします.マウスの動きに合わせてディスプレイ上でマウスカーソルを移動させることができますし，マウスに付いている2つのボタンを押したり離したりすることにより，ソフトウェアとのやりとりが可能になります.

ここでは，BASICでマウスを使用するための命令や関数について説明します.

マウスカーソルの移動範囲は特に指定しない限り，次に示すオリジナルスクリーン座標系の範囲内です.

| モード名 | 解像度 |
|---|---|
| ノーマルモードの高解像度カラー／白黒モード | 640×400ドット |
| ノーマルモードのカラー／白黒モード | 640×200ドット |
| ハイレゾモードのカラー／白黒モード | 1120×936ドット |

BASICは次のようなマウス機能を制御します.

- **MOUSE命令による制御**

  マウスの初期化

  マウスカーソルの表示位置の設定と表示／非表示の制御

  マウスカーソルの形状設定

  マウスカーソルの移動速度の設定

  マウスカーソルの移動範囲の設定

  マウスカーソルの色指定

  マウスの解放

- **MOUSE関数による情報取得**

  マウスカーソルの現在の座標を得る

  マウスボタンの状態を得る

マウスボタンの押離の回数を得る

マウスの移動距離を得る

・ON　MOUSE　GOSUB, MOUSE　ON/OFF/STOP 命令による制御

マウス割り込みの制御

## 9.2　マウス制御命令

MOUSE 命令に 0〜6 の範囲のパラメータをつけて実行することにより，マウスの各種機能を制御することができます．

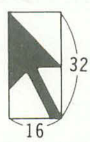
●マウスの初期化

```
MOUSE　0
```

BASIC でマウスの使用を宣言し，動作環境を初期化します．

MOUSE　0 を実行せずに他の MOUSE 関連の命令を実行すると，"MOUSE not initialized" エラーになります．

マウスカーソルは次のように初期化されます．

| 動作環境 | ノーマルモード | ハイレゾモード |
|---|---|---|
| 移動範囲 | 画面全体<br>(0,0)－(639,399(199)) | 画面全体<br>(0,0)－(1119,935) |
| 表示位置 | ディスプレイ画面の中心<br>(320,200(100)) | ディスプレイ画面の中心<br>(560,308) |
| 形 |  | |
| 表示画面 | 現在のアクティブページ | |
| カーソルの表示 | しない | |
| 色 | パレット番号 4 の色 | |
| 指示点 | (0, 0) | |
| 移動比率 | 水平方向　8<br>垂直方向　8 | |

●マウスカーソルの表示位置の設定と表示／非表示の制御

MOUSE　1〔,〈Sx〉〕〔,〈Sy〉〕〔,〈カーソルスイッチ〉〕

マウスカーソルの表示位置と表示／非表示を指定します．

スクリーン座標上の(Sx, Sy)にマウスカーソルの表示位置を指定します．〈Sx〉または〈Sy〉が省略された場合，現在のマウスカーソルのX座標(水平座標)またはY座標(垂直座標)が採用されます．

〈カーソルスイッチ〉は1を指定するとカーソルを表示し，0を指定すると表示しません．〈カーソルスイッチ〉を省略した場合，現在のカーソルの状態を引き継ぎます．

例1)　MOUSE　1,,, 1

マウスカーソルを現在の表示位置に表示します．MOSUE　0実行直後の場合，画面中央にマウスカーソルを表示します．マウスを移動させるとマウスカーソルはこの位置から動き始めます．

例2)　MOUSE　1, 100, 100, 1

マウスカーソルを(100, 100)の位置に表示します．

例3)　MOUSE　1,,, 0

現在表示されているマウスカーソルの表示をやめます．表示位置の座標は記憶されています．

●マウスカーソルの形状設定

ノーマルモードⓃ

MOUSE　2,〈X指示点〉,〈Y指示点〉,〈XOR文字列〉

ハイレゾモードⒽ

MOUSE　2,〈X指示点〉,〈Y指示点〉,〈XOR文字列〉〔,〈AND文字列〉〕

マウスカーソルの形状および指示点を指定します．マウスカーソルは決められた範囲内で任意の形状を定義することができます．また指示点はこの範囲内の任意の1点(1ドット)を選択でき，マウスカーソルの表示位置はこの指示点に対するものとなります．MOUSE関数で得られるマウスカーソルの表示位置はこの指示点の画面上の座標です．

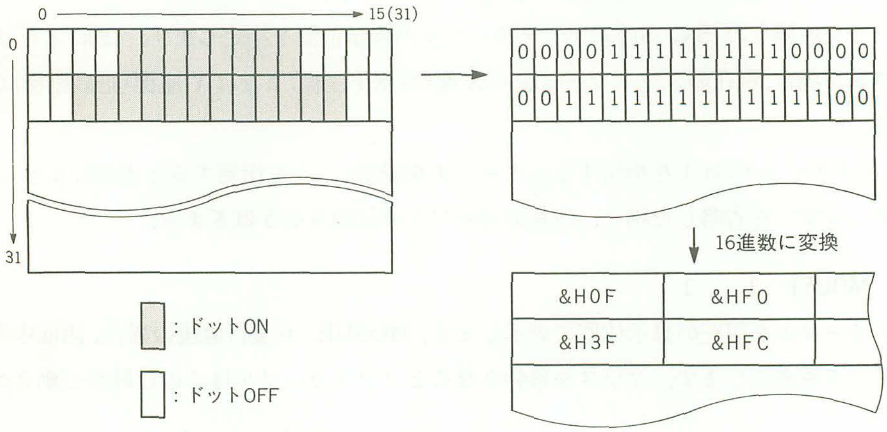
---

注意：〈AND〉文字列は，ハイレゾモード時のみ指定可能です．Ⓗ

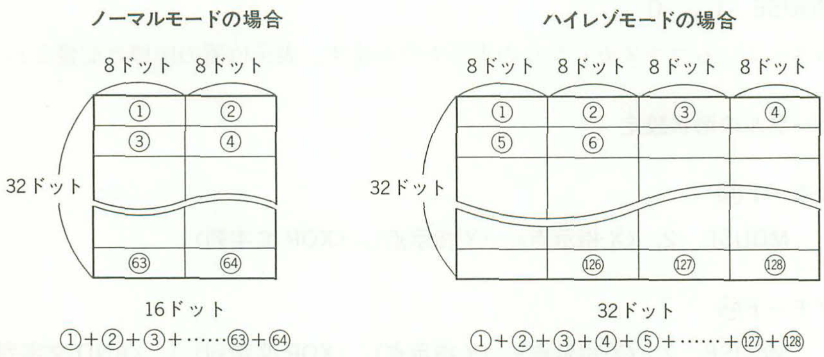
---

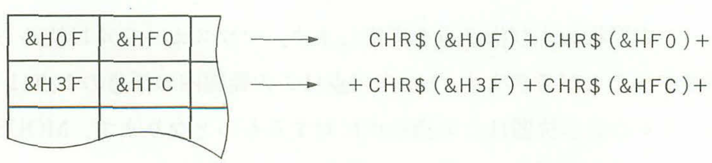
次に，マウスカーソルの形状を決定する方法を示します．

①マウスカーソルのドットのON/OFFを16進数に変換します．




②16進数に変換したデータを次の順序で連結し，文字列にします．



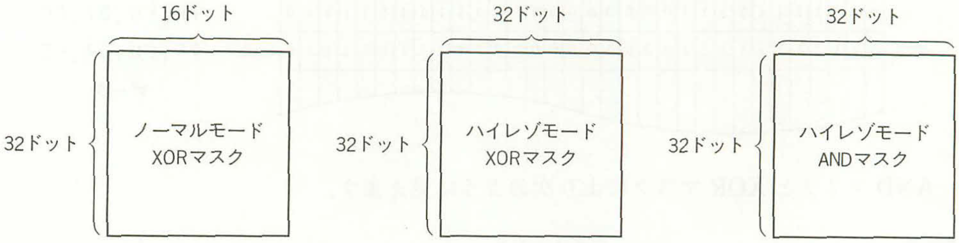




③連結した文字列を〈XOR文字列〉または〈AND文字列〉あるいはその両方に指定します．
次の例では〈XOR文字列〉だけを指定しています．

MOUSE　2，〈X指示点〉，〈Y指示点〉，CHR$(&H0F)+CHR$(&HF0)+………

マウスカーソルの大きさと指示点の指定範囲は次のとおりです．

| モード | マウスカーソルの大きさ | X指示点 | Y指示点 |
|---|---|---|---|
| ノーマルモード | 16×32ドット | 0〜15 | 0〜31 |
| ハイレゾモード | 32×32ドット | 0〜32 | 0〜32 |




〈XOR文字列〉だけを指定した場合，マウスカーソルの色はグラフィック情報(地の色)に影響されます．この場合，マウスカーソルがグラフィック情報(たとえばペイントした円など)と重なると，重なった部分の色が変化します．

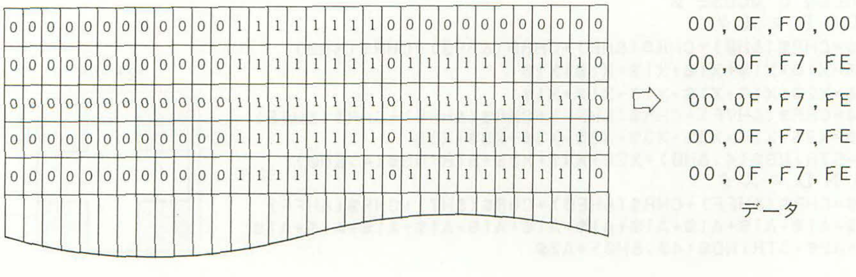
〈XOR文字列〉と〈AND文字列〉を指定すると，〈XOR文字列〉だけのときとは違い，グラフィック情報(地の色)に影響されないマウスカーソルを表示させることができます．〈AND文字列〉の指定は〈XOR文字列〉の場合と同じです．

〈AND文字列〉は省略することができます．MOSUE 0実行後のマウスカーソルはXORマスクだけで表示されます．

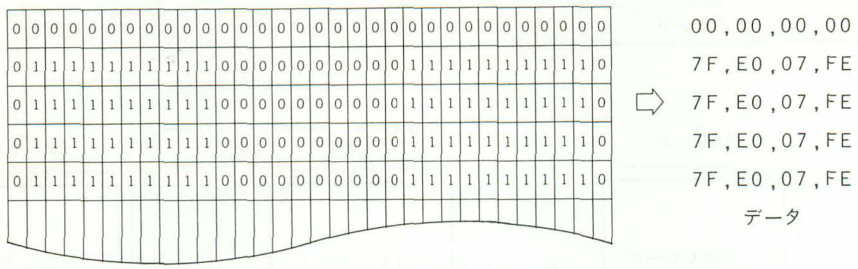
XORマスクとANDマスクのドットの状態で，画面には次のように描画されます．

| ANDマスク | XORマスク | 画面上のカーソルの状態 |
|---|---|---|
| 1 | 0 | 地の色 |
| 1 | 1 | 地のパレット番号を反転させたパレット番号の色 |
| 0 | 0 | パレット番号0の色 |
| 0 | 1 | パレット番号15の色 |

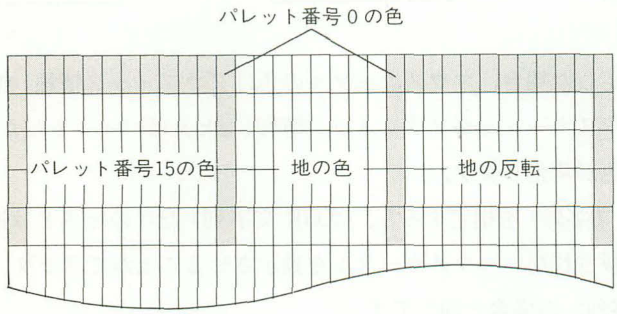
例)　ANDマスク



例)　XORマスク




例)　ANDマスクとXORマスクにより次のように見えます.

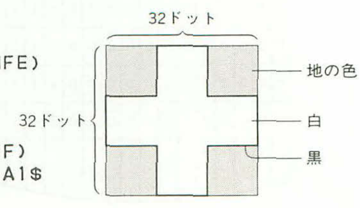



〈AND文字列〉= CHR\$(&H0) + CHR\$(&H0F) + CHR\$(&HF0) + CHR\$(&H00)
+ CHR\$(&H0) + CHR\$(&H0F) + CHR\$(&HF7) + CHR\$(&HFE)
+ CHR\$(&H0) + CHR\$(&H0F) + CHR\$(&HF7) + CHR\$(&HFE)
∫

〈XOR文字列〉= STRING\$(4, &H0)
+ CHR\$(&H7F) + CHR\$(&HE0) + CHR\$(&H07) + CHR\$(&HFE)
+ CHR\$(&H7F) + CHR\$(&HE0) + CHR\$(&H07) + CHR\$(&HFE)
∫

例)　次のプログラムを実行すると十字の白いカーソルが表示されます. Ⓗ

```
100 SCREEN 0:MOUSE 0
110 'XORマスク
120 X1$=CHR$(&H0)+CHR$(&HF)+CHR$(&HF0)+CHR$(&H0)
130 X2$=X1$+X1$+X1$+X1$+X1$+X1$
140 X2$=X2$+X1$+X1$+X1$+X1$+X1$
150 X3$=CHR$(&H7F)+CHR$(&HFF)+CHR$(&HFF)+CHR$(&HFE)
160 X4$=X3$+X3$+X3$+X3$+X3$+X3$+X3$+X3$
170 X$=STRING$(4,&H0)+X2$+X4$+X2$+STRING$(4,&H0)
180 'ANDマスク
190 A1$=CHR$(&HFF)+CHR$(&HE0)+CHR$(&H7)+CHR$(&HFF)
200 A2$=A1$+A1$+A1$+A1$+A1$+A1$+A1$+A1$+A1$+A1$+A1$
210 A$=A2$+STRING$(40,&H0)+A2$
220 '
230 COLOR=(15,&HFFFF)
240 MOUSE 2,15,15,X$,A$:MOUSE 1,,,1
```



●マウスカーソルの移動速度の設定

MOUSE 3, 〈方向〉, 〈移動比率〉

マウスカーソルの移動速度を決めます．移動速度とはマウスカーソルの移動距離に対するマウスの移動距離の割合です．

〈移動比率〉にはマウスカーソルを8ドット移動させるのに必要なマウスの移動距離を指定します．MOSUE 0実行直後の場合，X方向とY方向ともに8(標準値)に設定されます．〈移動比率〉には0〜32767の範囲の整数を指定します．0を指定すると指定された方向の移動比率は8(標準値)にもどります．

〈方向〉は，移動比率を変える方向を示します．0を指定するとX(水平)方向，1を指定するとY(垂直)方向の移動比率を変えます．

例1） MOUSE 3, 0, 4

マウスの標準の1/2の距離(すなわち4)の移動に対してマウスカーソルを水平方向に8ドット移動させます．つまり，マウスカーソルの水平方向の移動が標準よりも2倍速くなります．

例2） MOUSE 3, 1, 16

マウスの標準の2倍の距離(すなわち16)の移動に対してマウスカーソルを垂直方向に8ドット移動させます．つまり，マウスカーソルの垂直方向の移動が標準よりも2倍遅くなります．

●マウスカーソルの移動範囲の設定

MOUSE 4, 〈Sx1〉, 〈Sy1〉, 〈Sx2〉, 〈Sy2〉

マウスカーソルはグラフィック画面全体を移動させることができますが，移動範囲を画面上の一定範囲内に設定することができます．

移動範囲は(Sx1, Sy1)と(Sx2, Sy2)の2点を対角とする四角形の領域で表します．

例） MOUSE 4, 0, 0, 100, 100

マウスカーソルの移動範囲を画面左上端の(0, 0)と(100, 100)を対角とする領域内に設定します．

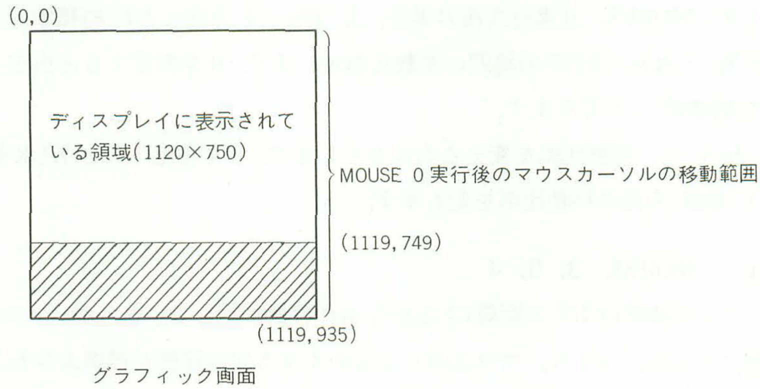
マウスカーソルが移動範囲の境界まで動いた後，マウスをいくら境界の外側の方向に動かしてもマウスカーソルは境界のところで止まったままです．

また，MOUSE 1で移動範囲外の座標を指定した場合，マウスカーソルは移動範囲内でMOUSE 1で指定された座標に最も近い位置に表示されます．

**・ハイレゾモードの場合の注意点Ⓗ**

マウスカーソルの移動範囲の最大値はグラフィック画面全体ですが，ハイレゾモードの場合，グラフィック画面(1120×936)と実際に表示されるディスプレイ画面(1120×750)はサイズが異なります．

たとえば，起動時のグラフィック画面とディスプレイ画面の関係を次に示します．




したがって，前図の斜線部分にマウスカーソルが移動すると，マウスカーソルが見えなくなります．斜線の範囲((0, 749)－(1119, 935))をディスプレイに表示させるには，SCROLL 命令を実行します．

▶参照　グラフィック画面とディスプレイ画面の関係→ 5.2 の「ハイレゾモードのグラフィック画面表示について」

**●マウスカーソルの色指定**

MOUSE　5,〈色〉

マウスカーソルの色を指定します．

〈色〉は 0 から 3 の値で指定し，それぞれ次のパレットに対応しています．

| 〈色〉 | 対応するパレット番号 | 初期状態のパレットの色 |
|---|---|---|
| 0 | パレット番号 1 | 青 |
| 1 | パレット番号 2 | 赤 |
| 2 | パレット番号 4 | 緑 |
| 3 | パレット番号 8 | 灰色＊ |

＊：ノーマルモード・4096 色中 16 色モードおよびハイレゾモードの場合のみ使用可能

マウスカーソルは，パレット番号に設定されている色に変わります．

ただし，8 色中・8 色および 4096 色中・8 色モードの時に 3 を指定するとエラーになります．

▶**参照**　パレットモード→6.3の「パレットモード」

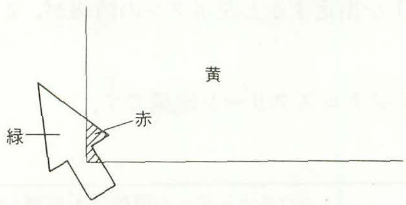
なお，パレット番号と実際の色の対応付けはCOLOR命令で行います．

例）COLOR(2, &HFFFF)……8色中・8色モードの場合はCOLOR(2,7)
　　MOUSE　5, 1

パレット番号2に白を設定し，マウスカーソルを白で表示させます．

**・マウスカーソルをXOR文字列だけで定義している場合の注意点**

マウスカーソルとグラフィック情報の間にはXORの関係が保たれています．このため，マウスカーソルがグラフィック情報と重なると，重なった部分の色が変化します．たとえば，次の例のように黄色で塗られたグラフィック情報と緑色のカーソルが重なると，重なった部分の色は赤になります(ただし，パレットが初期状態であるとします)．




これは次の関係によります．

| | | G | R | B |
|---|---|---|---|---|
| | 黄 | 1 | 1 | 0 |
| XOR ) | 緑 | 1 | 0 | 0 |
| | 赤 | 0 | 1 | 0 |

ただし，マウスカーソルが移動した後にグラフィック情報はもとの色にもどります．

**・マウスカーソルをXOR文字列，AND文字列で定義している場合の注意点Ⓗ**

MOUSE　5はXORマスクだけで構成されているマウスカーソルの色を指定する命令ですから，XORマスクとANDマスクの両方で構成されているマウスカーソルの色を指定することはできません．この場合のマウスカーソルの色は，MOUSE　2で決定されます．

なお，XORマスクとANDマスクで構成されているマウスカーソルに対してMOUSE　5を実行すると，そのマウスカーソルは指定された色に変化して，それ以降XORマスクだけの表示になってしまいますので注意してください．

**●マウスの解放**

```
MOUSE 6
```

マウスカーソルの表示をやめ，マウス機能をすべてOFFにします．マウスの使用を終了する場合，必ずこの命令を実行してください．

MOUSE　6を実行後に再びマウスを使用する場合は，MOUSE　0を実行してマウスを初期化する必要があります．

## 9.3 マウス関数

MOUSE関数を使用して，マウスボタンの状態やマウスカーソルの座標を得ることができます．

次の表の〈ボタン番号〉に1を指定すると左ボタンの情報が，2を指定すると右ボタンの情報が得られます．

得られる座標はすべてオリジナルスクリーン座標です．

| MOUSE(0) | マウスカーソルの現在のX座標(水平座標)を得る |
|---|---|
| MOUSE(1) | マウスカーソルの現在のY座標(垂直座標)を得る |
| MOUSE(2,〈ボタン番号〉) | 〈ボタン番号〉で指定されたボタンの状態を得る<br>ボタンが押されているときは1を，離されているときは0を得る |
| MOUSE(3,〈ボタン番号〉) | 最後にこの関数が実行されてから現在までの間に〈ボタン番号〉で指定されたボタンが押された回数を得る |
| MOUSE(4,〈ボタン番号〉) | 〈ボタン番号〉のボタンが最後に押されたときのX座標を得る |
| MOUSE(5,〈ボタン番号〉) | 〈ボタン番号〉のボタンが最後に押されたときのY座標を得る |
| MOUSE(6,〈ボタン番号〉) | 最後にこの関数が実行されてから現在までの間に〈ボタン番号〉で指定されたボタンが離された回数を得る |
| MOUSE(7,〈ボタン番号〉) | 〈ボタン番号〉のボタンが最後に離されたときのX座標を得る |
| MOUSE(8,〈ボタン番号〉) | 〈ボタン番号〉のボタンが最後に離されたときのY座標を得る |
| MOUSE(9) | 最後にこの関数が実行されてから，現在までにマウスカーソルが移動したX方向の距離を得る<br>右向きが正の方向で，左向きが負の方向 |
| MOUSE(10) | 最後にこの関数が実行されてから，現在までにマウスカーソルが移動したY方向の距離を得る<br>下向きが正の方向で，上向きが負の方向 |

# 9.4 マウス割り込み

マウス割り込み制御命令は，マウスの事象に対する割り込み命令です．これを利用することにより，マウスの事象(マウスが移動した，マウスのボタンが押されたなど)を動的に管理することができます．

マウス割り込み制御命令は次のとおりです．

ON　MOUSE　GOSUB 命令
MOUSE　ON/OFF/STOP 命令

●割り込み処理ルーチンの開始行の定義

ON MOUSE GOSUB 命令

ON　MOUSE (〈事象番号〉)　GOSUB　〈行番号〉

マウスの種々の動作による割り込みが発生したときに分岐する処理ルーチンを定義します．

〈事象番号〉は 1～5 の値で指定します．内容は次のとおりです．

| 事象番号 | 事　　象 |
|---|---|
| 1 | マウスが移動した |
| 2 | 左側のボタンが押された |
| 3 | 右側のボタンが押された |
| 4 | 左側のボタンが離された |
| 5 | 右側のボタンが離された |

〈行番号〉は指定した事象による割り込みが発生したときに分岐する処理ルーチンです．

割り込み処理ルーチンへの分岐が起こるのは，対応する事象が割り込み許可状態になっている場合です(MOUSE　ON 命令)．

割り込み処理ルーチンに制御が移っている間は対応する事象に対する割り込みは自動的に停止状態になります．

割り込み処理状態からの復帰は RETURN 命令で行います．

▶参照　ON　MOUSE　GOSUBE 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

**●割り込み事象の制御**

**MOUSE　ON/OFF/STOP 命令**

MOUSE (〈事象番号〉) { ON / OFF / STOP }

マウスの種々の動作による割り込みを許可(ON)するか，禁止(OFF)するか，停止(STOP)するかを宣言します．

〈事象番号〉は1～5の値で指定します．内容は次のとおりです．

| 事象番号 | 事　象 |
|---|---|
| 1 | マウスが移動した |
| 2 | 左側のボタンが押された |
| 3 | 右側のボタンが押された |
| 4 | 左側のボタンが離された |
| 5 | 右側のボタンが離された |

▶参照　MOUSE　ON/OFF/STOP 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

## 9.5 マウス使用上の注意事項

・マウスを利用しながら LINE 命令や CIRCLE 命令などグラフィック命令を実行する場合は，グラフィック命令を実行する直前にマウスカーソルを消し，命令実行後に再びマウスカーソルを表示するようにしてください．

---

**参考**：MOUSE　1,,, 0……マウスカーソルを消します．
　　MOUSE　1,,, 1……マウスカーソルを表示します．

---

・マウスカーソルが表示されたままでマウスカーソルにグラフィック情報が重なると，マウスカーソルが移動した後，マウスカーソルとグラフィック情報が重なっていたところの色が変わることがあります．

・白黒モードでマウスカーソルの色指定(MOUSE　5)を実行すると，指定された色に対応するページにマウスカーソルが書き込まれます．

・指定した色に対応するページを表示させるには，SCREEN 命令を実行して対応するディスプレイページを表示するモードにしてください．

▶参照　ディスプレイページ→5.2の「ページ」
SCREEN 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

## 9.6 応用プログラム

このマウスのサンプルプログラムは，左ボタンを押している間マウスカーソルと同じ色でマウスカーソルの軌跡を描くプログラムです．マウスカーソルの色は最初緑で，右ボタンを押すと，青→赤→緑と変化していき，さらにもう一度押すとプログラムは終了します．

次に，このプログラムでマウスに関する命令／機能について解説します．

160行のMOUSE　0でマウスの諸機能の初期化を行います．MOUSE　0は，マウス機能を使用するときに，最初に実行しなければならない命令です．

170行から190行までは，マウスによる割り込みルーチンの開始行を定義し，割り込みを許可するものです．170行は，左ボタンが押されたら＊DRAWLINEに分岐することを定義しています．180行は，右ボタンが押されたら＊CHANGEMODEに分岐することを定義しています．190行は，左ボタンが離されたら，＊STOPDRAWに分岐することを定義しています．

200行でマウスカーソルの表示を始めます．

210行から230行までは，割り込みが発生するまで現在のマウスカーソルの位置(X座標とY座標)を画面の左上に表示します．

260行のMOUSE　6でマウスの諸機能を終了させます．MOUSE　6はマウス機能の使用を終了するときに実行しなければならない命令です．

320行から370行までは，左ボタンが押されている間，マウスカーソルの軌跡をマウスカーソルと同じ色で描くサブルーチンです．360行では，LINE命令の前でマウスカーソルの表示を停止し，LINE命令の後でマウスカーソルの表示を再開しています．これは，マウスカーソル表示中にグラフィック命令を実行して画面が乱れることがないようにするためです．

390行から490行までは，左ボタンが離されたときに線を描くのをやめ，さらにマウスカーソルの色を順次変えて行くためのサブルーチンです．このプログラムは右ボタンを4回押すと終了するようになっています．

510行から550行までは，現在のマウスカーソルの座標を画面の左上に実際に表示するサブルーチンです．

```
100 ' ****************************************
110 '*    マウスのサンプルプログラム    *
120 ' ****************************************
130 CLS 3:C=4
140 LOCATE Ø,2:PRINT "右ボタンを押すとカーソルが青になります"
150 LOCATE Ø,1:PRINT "左ボタンを押すと線が引けます"
160 MOUSE Ø
170 ON MOUSE(2) GOSUB *DRAWLINE    :MOUSE(2) ON
180 ON MOUSE(3) GOSUB *CHANGEMODE :MOUSE(3) ON
190 ON MOUSE(4) GOSUB *STOPDRAW    :MOUSE(4) ON
200 MOUSE 1,,,1
210 LOCATE 1Ø,1:PRINT "押すと線が引けます"
220 GOSUB *DISPLAYCOORDINATE
230 GOTO 21Ø
240 '
250 *MOUSEEND
260 MOUSE 6
270 CLS 3:END
280 '
290 ' ******************************************
300 '*    このプログラムのサブルーチン    *
310 ' ******************************************
320 *DRAWLINE
330 FLAG=Ø
340 LOCATE 1Ø,1:PRINT "離すと元に戻ります"
350 POINT(MOUSE(Ø),MOUSE(1))
360 MOUSE 1,,,Ø:LINE -(MOUSE(Ø),MOUSE(1)),C:MOUSE 1,,,1
370 IF FLAG=Ø THEN 36Ø ELSE RETURN 22Ø
380 '*************************************
390 *STOPDRAW
400 FLAG=-1:RETURN
410 '*************************************
420 *CHANGEMODE
430 IF MODE=3 THEN RETURN *MOUSEEND
440 MOUSE 5,MODE:MODE=MODE+1
450 LOCATE 26,2
460 ON MODE GOTO 47Ø,48Ø,49Ø
470 PRINT "赤":C=1:RETURN
480 PRINT "緑":C=2:RETURN
490 PRINT "消えてプログラムが終了します":C=4:RETURN
500 '*************************************
510 *DISPLAYCOORDINATE
520 MOUSE(3) STOP
530 LOCATE Ø,Ø:PRINT "X=";MOUSE(Ø),"Y=";MOUSE(1)
540 MOUSE(3) ON
550 RETURN
```

# 第10章

# 機械語プログラム

$N_{88}$-日本語 BASIC(86)(MS-DOS 版)では，機械語プログラムをメモリ上に書き込んでおき，それを BASIC から呼び出して実行させることができます．機械語プログラムを実行するためには BASIC の命令を使って次に示すような準備をしなければなりません．この章では機械語プログラムを BASIC から使う場合の準備と実行方法の流れを解説します．

なお，アセンブラなどを使って機械語プログラムそのものを作成する方法はここでは解説しませんので，機械語の入門書などを参考にしてください．

**機械語プログラムが使用するメモリ領域の確保**

CLEAR 命令

↓

**使用するセグメントの宣言**

DEF　SEG 命令と SEGPTR 関数

↓

**機械語プログラムをメモリに置く** ←→ **機械語プログラムをディスクにセーブする**

POKE 命令　　　　BSAVE 命令

BLOAD 命令

↓

**BASIC から機械語プログラムを呼び実行させる**

DEF　USR 命令と USR 関数

CALL 命令

BLOAD 命令

# 10.1　メモリレイアウトとセグメント

機械語では00000H番地からFFFFFH番地までの最大1Mバイトのメモリを扱うことができます．

しかし，番地の指定の際には00000H番地からFFFFFH番地までの絶対アドレスを指定することはできません．メモリを64Kバイト単位のセグメントという単位に分けて扱います．番地の指定の前には，あらかじめセグメントの先頭番地を宣言し，このセグメントの先頭番地(セグメントベース)からの相対アドレスで指定します．

## ■セグメントベース

セグメントの先頭番地を宣言するにはDEF　SEG命令を使います．

DEF　SEG命令

　　DEF　SEG=<セグメントベース>

セグメントの先頭番地の指定は実際の物理アドレス(絶対アドレス)を16で割った値で行います．これをセグメントベースといいます．たとえば，機械語プログラムをロードする場合，まずセグメントを機械語プログラム領域の先頭番地(セグメントベース)として宣言する必要がありますが，このセグメントはSEGPTR(2)によって得ることができますから，

　　DEF　SEG=SEGPTR(2)

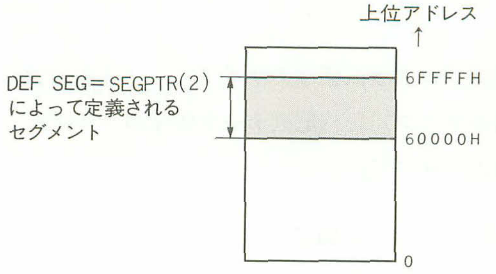
と指定します．この宣言をすると，セグメントは機械語プログラム領域の先頭番地から始まる64Kバイト分のメモリということになります．

---

**注意**：セグメントベースは物理アドレスを16で割った値ですから，セグメントの先頭番地の物理アドレスは必ず16の倍数になります．

---

セグメントの設定後，そのセグメント内の特定番地を指定するには相対アドレスを使用します．相対アドレスの指定可能範囲は0000H〜FFFFHです．これにより，セグメントベースにより決定されたセグメントの先頭番地から64Kバイト分のメモリ中の任意のアドレスをアクセスすることができます．前の例で，SEGPTR(2)の値が6000Hであるとすると，60000H〜6FFFFH番地が指定できることになります．




〔物理アドレス〕

次の命令は，相対アドレスを利用するため，実行前に DEF　SEG 命令によるセグメントの指定が必要です．

CALL，BLOAD，BSAVE，DEF　USR，PEEK，POKE

セグメントの概念は機械語を使う上でしっかり理解する必要があります．以下の解説では実際の命令のなかでセグメントがどのように使われているのか注意してください．

▶参照　セグメント→「第 14 章　メモリマップとメモリ管理」

## 10.2 機械語プログラムの準備

### ■機械語プログラムが使用するメモリ領域の確保

機械語プログラムを使う場合にまず最初にやらなければならないことは，機械語プログラム用のメモリ領域と BASIC プログラム用のメモリ領域を別に確保することです．もし両者のメモリ領域が重複すると，BASIC の動作が異常になったり，機械語プログラムが失われてしまうことになります．

機械語プログラムのメモリ領域と BASIC のメモリ領域の境界の宣言，つまり機械語プログラムのメモリ領域の確保は CLEAR 命令で行います．この命令は DEF　SEG 命令より先に実行しなければなりません．

CLEAR 命令

インタプリタの場合

CLEAR　〔〈機械語プログラム領域のサイズ〉〕〔，〈利用者スタック領域のサイズ〉〕
〔，〈配列データ領域のサイズ〉〕〔，〈チャイルドプロセス領域のサイズ〉〕

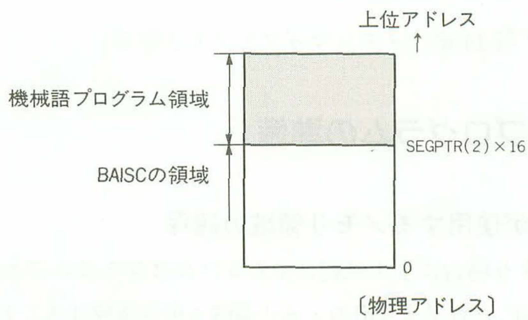
コンパイラの場合

CLEAR　〔〈機械語プログラム領域のサイズ〉〕〔,〈文字列データ領域およびスタック領域のサイズ〉〕〔,〈配列データ領域のサイズ〉〕〔,〈チャイルドプロセス領域のサイズ〉〕

最初のパラメータ〈機械語プログラム領域のサイズ〉で，機械語プログラム用のメモリ領域の確保を行います．メモリ上では，〈機械語プログラム領域のサイズ〉で指定した値の16倍の大きさ(バイト)の領域が，SEGPTR(2)で示されるセグメントベースを先頭として確保されます．したがって，たとえば，

```
CLEAR　&HE00
```

とすると物理アドレスSEGPTR(2)×16番地〜SEGPTR(2)×16+E00H番地の範囲が機械語プログラムで使用できるメモリ領域となります．




## ■使用するセグメントの宣言

CLEAR命令による機械語のメモリ領域の確保が終わったら，実際に機械語プログラムをメモリ上に書き込むための準備として，DEF　SEG命令を使ってセグメントを宣言します．

CLEAR命令の例に合わせた例をあげると，次のようになります．

```
DEF　SEG=SEGPTR(2)
```

この命令を実行するとDEF　SEG命令を使ってセグメントを変更しない限り，メモリへの読み書きはこのセグメントが対象になります．

## ■機械語プログラムの格納

機械語プログラムを実行するには，まずプログラムをメモリ上に格納(ロード)しなければなりません．そのための方法として次のような2つの方法があります．

### ● POKE 命令を使う方法

POKE 命令でプログラムを書き込む方法です.

POKE 命令は，データを直接メモリ上に書き込むための命令です.

POKE 命令

POKE　〈アドレス〉,〈式〉

POKE 命令は，〈アドレス〉に指定した番地に〈式〉の値を書き込みます．〈アドレス〉には，先に実行した DEF　SEG 命令で宣言された，現在のセグメントベース(SEGPTR(2)の値)からの相対アドレスを指定します．比較的短いプログラムなら，DATA 行で機械語プログラムを用意し，READ 命令で読み取り，POKE 命令でメモリに書き込むことができます.

### ● BLOAD 命令を使う方法

BLOAD 命令は，ディスクにすでにファイルとしてセーブされた機械語プログラムをメモリ上にロードするときに使います.

BLOAD 命令

BLOAD　〈ファイルディスクリプタ〉〔,　〈ロードアドレス〉〕〔, R〕

BLOAD 命令は，〈ファイルディスクリプタ〉で指定されたファイル名の機械語プログラムをメモリ上にロードします．〈ロードアドレス〉には，機械語ファイルを読み込む先頭の番地を，先に実行した DEF　SEG 命令で宣言された，現在のセグメントベース(SEGPTR(2)の値)からの相対アドレスで指定します．〈ロードアドレス〉を省略した場合には，現在のセグメントベースに，機械語プログラムをセーブする際に BSAVE で指定した開始アドレス(相対アドレス)を加えた番地からロードが行われます.

▶参照　R オプション→10.3 の「BLOAD 命令を使った機械語プログラムの実行」

なお，BLOAD 命令を使うには，あらかじめ機械語プログラムがディスクにセーブされていなければなりません．セーブには基本的に 2 つの方法があります.

### ● BSAVE 命令を使う方法

BSAVE 命令

BSAVE　〈ファイルディスクリプタ〉,　〈開始アドレス〉,　〈長さ〉

BSAVE 命令は，メモリ上の機械語プログラムを〈ファイルディスクリプタ〉で指定されたファイル名でディスクにセーブします．〈開始アドレス〉には，セーブを開始する番地を，現在のセグメントベースからの相対アドレスで指定します．〈長さ〉にはセーブする機械語プログラムのバイト数を指定します.

●アセンブラ／リンカを使う方法

MS-DOS上で動作するアセンブラおよびリンカを使用して作成したEXE形式のファイルを，MS-DOSのEXE2BINコマンドでCOM形式のファイルに変換する方法です．こうして作られたCOM形式ファイルは，機械プログラムとしてBLOAD命令でメモリにロードすることができます。

詳細については，MS-DOSのアセンブラ，リンカの説明をご覧ください．

## 10.3　機械語プログラムの実行

BASICから機械語プログラムを使う方法は3つあります．

・USR関数を使う
・CALL命令を使う
・BLOAD命令を使う

これらの方法について，以降の項でそれぞれ別に解説します．

### ■ USR関数を使った機械語プログラムの実行

USR関数は，メモリ上に置かれた機械語プログラムを，関数の形で実行するものです．

USR関数を使用するには，まず機械語プログラムの実行開始アドレスをUSR関数そのものにセットしなくてはなりません．これを機械語関数の定義といい，DEF　USR命令を用いて行います．

DEF　USR命令

DEF　USR[〈番号〉]＝〈開始アドレス〉

〈番号〉には0から9までの数値を指定することができます．〈開始アドレス〉には機械語プログラムの開始アドレスを相対アドレス(直前のDEF　SEG命令で定義したセグメントベースからの相対アドレス)で指定します．

たとえば次のようにすると，

```
DEF　USR0＝&HD000
```

セグメントベースからの相対アドレスD000Hのところから始まる機械語プログラムが0番の機械語関数，すなわちUSR0として定義されます．このようにして，最大10個の機械語関数(USR0～USR9)を定義して実行することができます．

なお，機械語関数の定義は，機械語プログラムをメモリへロードする前でも，また後でもかまいません．ロード前の場合には機械語プログラムをロードする番地を想定して，定義を行い

ます．

DEF　USR命令は機械語関数の定義を行うだけです．実際に機械語プログラムを実行させるにはUSR関数を使います．

USR関数を次の書式で用いると機械語プログラムに制御が移ります．

**USR関数**

〈変数〉＝USR〔〈番号〉〕(〈引数〉)

〈番号〉にはDEF　USR命令で定義された機械語関数の番号を指定します．省略すると0を指定したのと同様になります．

〈引数〉には機械語プログラムに引き渡す変数，定数，式を指定します．〈引数〉は省略できませんので，機械語プログラムに値を引き渡す必要がない場合には何らかの値をダミーとして指定します．

〈変数〉には〈引数〉と同じ型の変数を指定してください．〈変数〉には機械語プログラムの実行後その実行結果が格納されます．

なお，USR関数は，BASIC上で実行されるだけでメモリ上の機械語プログラムに制御が移ります．すなわち，必ずしも"〈変数〉＝USRn(x)"の形で実行する必要はなく，たとえば"PRINT USRn(x)"としたり，他の関数や命令のパラメータとして使用したりしてもかまいません．

機械語プログラム側では，その実行結果を特定のメモリ領域に入れておくようにします．こうすると，機械語プログラムの実行が終了した際，〈変数〉に結果の値が入れられた状態で引き続きBASICのプログラムが実行されます．この特定のメモリ領域については次の項で説明します．

---

**注意**：機械語プログラムを終了したあとで制御をBASICにもどすためには，機械語プログラムの最後で必ず機械語のIRET命令を用いなければなりません．

---

**●機械語プログラムとBASICとの間でデータを交換するためのメモリ領域のアドレス**

BASICから渡された〈引数〉を機械語プログラム上で使用するには，〈引数〉の値がどこのメモリ領域に置かれているかを知る必要があります．また，機械語プログラムが実行されたあとでBASICに値を返すには，機械語側でどこのメモリ領域に値を入れればよいかを知る必要があります．

実際には，どちらの場合も同じアドレスを使用します．次にこのアドレスを実際に求める方法について説明します．

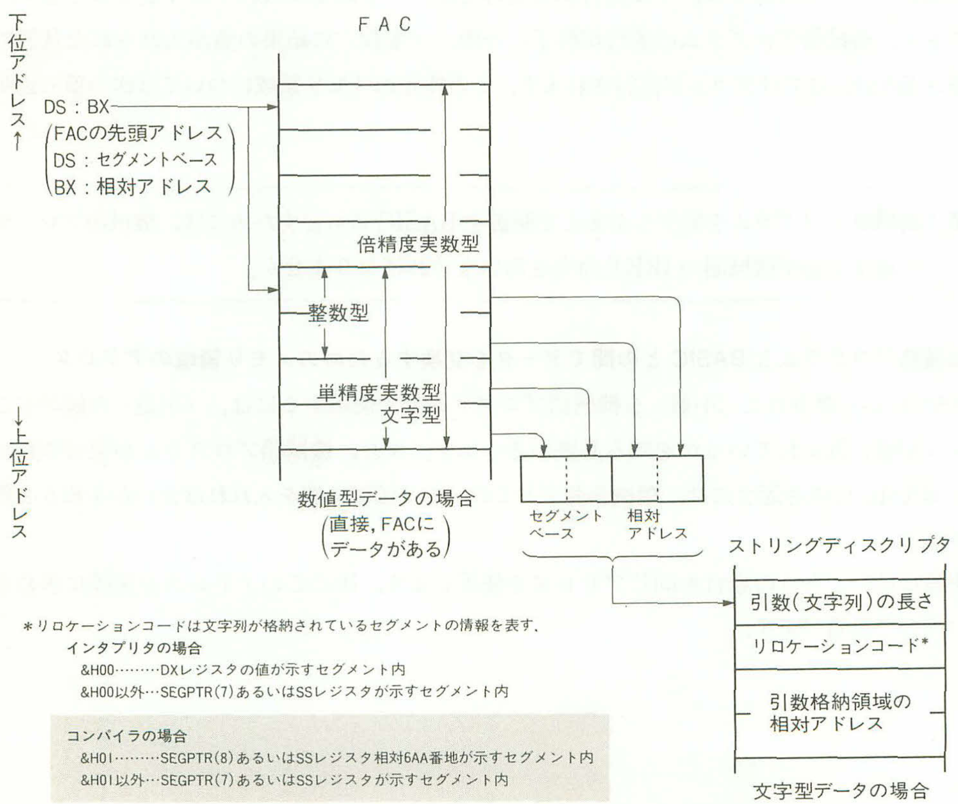
### ・〈引数〉の内容を機械語プログラムで使う場合

〈引数〉の内容が入るメモリのアドレスを示すデータは，USR関数が実行された時点でCPUのALレジスタ，DSレジスタ，BXレジスタに入れられます．したがって機械語プログラムでは，これらのレジスタから得たアドレスによって示されるメモリの内容を読み出すことにより，〈引数〉の値を得ることができます．

| 引数の型 | AL レジスタ | DS レジスタ | BX レジスタ |
|---|---|---|---|
| 整数型 | 2 | FACのあるテキストセグメントのセグメントベース(引数の内容が入っている領域のセグメントベース) | FACの5バイト目の相対アドレス(引数の内容が入っている領域のセグメントベースからの相対アドレス) |
| 単精度実数型 | 4 | | |
| 文字型 | 3 | | |
| 倍精度実数型 | 8 | | FACの1バイト目の相対アドレス(引数の内容が入っている領域のセグメントベースからの相対アドレス) |

表中，FACとは"浮動小数点アキュムレータ"のことで，BASICが算術演算などを行う際に使用する(8バイトの)作業領域を表します．

〈引数〉の内容は，その型によって表のようにFACへの格納のされ方が異なります．



文字型の場合は引数の値が格納されているわけではなくストリングディスクリプタのアドレスが格納されています．ストリングディスクリプタとは4バイトで構成される文字列(引数)のデータのことです．

**・機械語プログラムでBASICに値を返す場合**

機械語プログラムからBASICに値を返す際に使用されるメモリ領域は，USR関数の実行時に〈引数〉が格納された領域とまったく同じです．したがって，BASICにもどる際には，機械語プログラムで求められた値をFAC中に設定し，さらにALレジスタにその値の型を設定することが必要です．なお，返す値の型は，〈引数〉と同じでなければなりません．

**●機械語関数を使ったプログラム例**

機械語関数の使用例として，文字列中に含まれる小文字を大文字に変換して返す機械語プログラムを含んだBASICプログラムを示します．DATA行で機械語プログラムを用意しておき，POKE命令を使ってメモリに書き込んでいます．

機械語プログラムは，次に示すような手順で，与えられた引数自身を直接変更します．

①ALレジスタの値を調べて与えられた引数が文字型かどうかを調べ，文字型でない場合にはそのままBASICにもどる．

②引数が文字型なら，BXレジスタとDSレジスタで与えられたFACの内容からストリングディスクリプタの番地を求める．

③ストリングディスクリプタの番地を見て文字列の長さと文字列の置かれている番地を求める．

④文字列を最初から1文字ずつ調べ，小文字の時には5ビット目を0にすることにより大文字に変換する．

---

```
100 CLEAR &H10
110 MACHINE=&H0
120 DEF SEG=SEGPTR(2)
130 DEF USR0=MACHINE
140 '
150 FOR ADR=MACHINE TO MACHINE+48
160   READ BYTE:POKE ADR,BYTE 'poke ML program
170 NEXT ADR
180 '
190 INPUT A$
200   B$=USR0(A$)
210   PRINT B$
220 GOTO 190
230 END
240 '
250 '  machine language program data
260 '
270 DATA &h3c,&h03          :' CMP   AL,03H    ; string ?
280 DATA &h75,&h2c          :' JNZ   STRCNV90  ; jump no
290 DATA &h53               :' PUSH  BX
300 DATA &h51               :' PUSH  CX
310 DATA &h52               :' PUSH  DX
320 DATA &h56               :' PUSH  SI
```

```
330 DATA &hc5,&h1f              :' LDS    BX,[BX]
340 DATA &h8b,&h0f              :' MOV    CX,[BX]    ; CH:rel no.  CL:data length
350 DATA &h80,&hfd,&h00         :' CMP    CH,00H
360 DATA &h74,&h04              :' JZ     STRCNV10 ; jump yes
370 DATA &h32,&hed              :' XOR    CH,CH
380 DATA &h8c,&hd2              :' MOV    DX,SS
390 '                   STRCNV10:'
400 DATA &h8b,&h77,&h02         :' MOV    SI,2[BX] ; get offset
410 DATA &h8e,&hda              :' MOV    DS,DX
420 '                   STRCNV20:'
430 DATA &h8a,&h24              :' MOV    AH,[SI]
440 DATA &h80,&hfc,&h61         :' CMP    AH,61H     ; cmp. small "a"
450 DATA &h72,&h08              :' JC     STRCNV30 ; jump less
460 DATA &h80,&hfc,&h7b         :' CMP    AH,7BH     ; cmp. small "z"
470 DATA &h73,&h03              :' JNC    STRCNV30 ; jump greater or equal
480 DATA &h80,&h24,&hdf         :' AND    BYTE PTR [SI],0DFH      ; convert
490 '                   STRCNV30:'
500 DATA &h46                   :' INC    SI
510 DATA &he2,&hee              :' LOOP   STRCNV20
520 DATA &h5e                   :' POP    SI
530 DATA &h5a                   :' POP    DX
540 DATA &h59                   :' POP    CX
550 DATA &h5b                   :' POP    BX
560 '                   STRCNV90:'
570 DATA &hcf                   :' IRET
```

## ■CALL命令を使った機械語プログラムの実行

CALL命令を使って，機械語プログラムを呼び出すことができます．CALL命令は機械語関数を使う場合と違い，複数の引数を機械語プログラムに渡し，また複数の値をBASICに返すことができます．

### CALL命令

CALL　〈変数名〉〔(〈引数1〉，〈引数2〉，…，〈引数n〉)〕

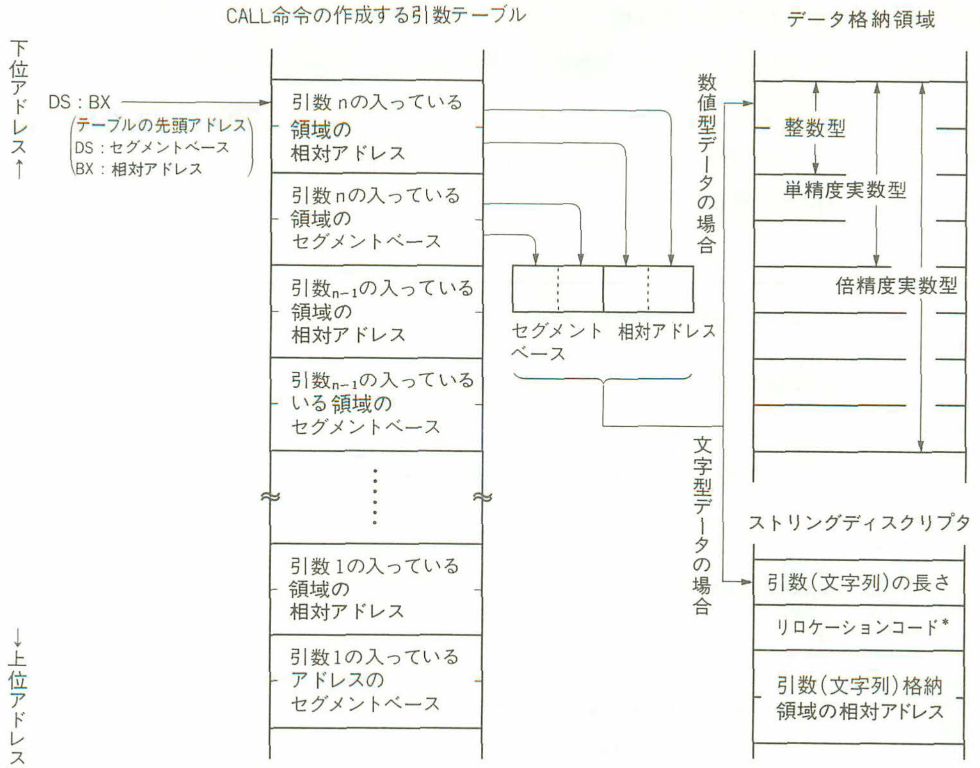
〈変数名〉には，機械語プログラムの実行開始アドレス(直前に実行したDEF　SEG命令で指定したセグメントベースからの相対アドレス)がまえもって代入されている変数名を指定します．

〈引数1〉～〈引数n〉にはすべての型の変数を任意の数だけ指定することができます．

各引数の内容が入っているメモリのアドレスは，CPUのDSレジスタ，BXレジスタを参照すればわかるようになっています．

DSレジスタとBXレジスタは，引数の内容が入っているメモリ領域のアドレスを直接指し示しているのではなく，引数の内容が入っているメモリ領域のアドレス情報を持っているテーブルのアドレスを示します．

アドレスの指し示し方を図解すると次のようになります．




＊リロケーションコードは文字列が格納されているセグメントの情報を表す．

インタプリタの場合
&H00………DXレジスタの値が示すセグメント内
&H00以外…SEGPTR(7)あるいはSSレジスタが示すセグメント内

コンパイラの場合
&H01………SEGPTR(8)あるいはSSレジスタ相対6AA番地が示すセグメント内
&H01以外…SEGPTR(7)あるいはSSレジスタが示すセグメント内

機械語プログラムからBASICに値を返す場合にもこのアドレスを使用します．すなわち，引数のデータを機械語で書き換えることにより，BASICにもどったときの引数(変数)の内容を変えるわけです．ただし，引数の型は，CALL命令を実行する前と後で同じでなければなりません．したがって，機械語プログラムに引き渡す引数とBASICに返す(受け渡す)引数の個数や型が異なる場合には，引き渡し専用または受け渡し専用の引数を指定するなどの対策を講じるようにしてください．

CALL命令による〈引数〉のやりとりで気をつけなければならないことは，引数の型とその個数が機械語プログラムに渡されないことです．機械語プログラムを作るときにはBASICとの整合性に注意してください．

---

**注意**：機械語プログラムを終了したあとで制御をBASICにもどすためには，機械語プログラムの最後で必ず機械語のIRET命令を用いなければなりません．

---

### ●CALL命令を使ったプログラム例

CALL命令の使用例として，2つの数のうち最大値を与える機械語プログラムを含んだBASICプログラムを示します．

機械語プログラムは，BXレジスタの示す引数テーブルからA％とB％の内容を取り出し，大きな方をC％の番地に格納します．

```
100 '   CALL statement sample program
110 '
120 CLEAR &H10
130 DEF SEG=SEGPTR(2)
140 MAX=&H0
150 '
160 FOR ADR=MAX TO MAX+47
170   READ BYTE:POKE ADR,BYTE   'poke ML language
180 NEXT ADR
190 '
200 C%=0
210 INPUT A%,B%
220   CALL MAX(A%,B%,C%)
230   PRINT C%
240 GOTO 210
250 END
260 '
270 '   machine language program data
280 '
290 '                       MAX:
300 DATA &h50                 :' PUSH  AX
310 DATA &h53                 :' PUSH  BX
320 DATA &h51                 :' PUSH  CX
330 DATA &h56                 :' PUSH  SI
340 DATA &h06                 :' PUSH  ES
350 DATA &h8b,&h4f,&h0a       :' MOV   CX,10[BX]      ; get 1st base addr.
360 DATA &h8e,&hc1            :' MOV   ES,CX
370 DATA &h8b,&h77,&h08       :' MOV   SI,8[BX]       ; get 1st offset addr.
380 DATA &h26,&h8b,&h04       :' MOV   AX,ES:[SI]     ; get 1st data
390 DATA &h8b,&h4f,&h06       :' MOV   CX,6[BX]       ; get 2nd base addr.
400 DATA &h8e,&hC1            :' MOV   ES,CX
410 DATA &h8b,&h77,&h04       :' MOV   SI,4[BX]       ; get 2nd offset addr.
420 DATA &h26,&h3b,&h04       :' CMP   AX,ES:[SI]     ; compare data
430 DATA &h7d,&h03            :' JGE   CMP10          ; jump 1st >= 2nd
440 DATA &h26,&h8b,&h04       :' MOV   AX,ES:[SI]     ; move 2nd to AX
450 '                     CMP10:'
460 DATA &h8b,&h4f,&h02       :' MOV   CX,2[BX]       ; get 3rd base addr.
470 DATA &h8e,&hc1            :' MOV   ES,CX
480 DATA &h8b,&h37            :' MOV   SI,0[BX]       ; get 3rd offset addr.
490 DATA &h26,&h89,&h04       :' MOV   ES:[SI],AX     ; move AX to 3rd area
500 DATA &h07                 :' POP   ES
510 DATA &h5e                 :' POP   SI
520 DATA &h59                 :' POP   CX
530 DATA &h5b                 :' POP   BX
540 DATA &h58                 :' POP   AX
550 DATA &hcf                 :' IRET
```

## ■ BLOAD 命令を使った機械語プログラムの実行

BLOAD 命令に R オプションを指定して実行すると，機械語プログラムのロード後ただちにその先頭番地から実行が開始されます．したがって指定する機械語プログラムのファイルは実行可能なプログラムでなければなりません．この場合には BASIC から引数を渡すことや BASIC に実行結果を返すことはできません．

# 第11章

# 拡張機能

## 11.1　概要

GP-IB制御機能，サウンド制御機能，BRANCH4670対応ネットワーク制御機能，MS-NETWORKS制御機能，電話制御機能および10進演算機能は，BASICの拡張機能として位置付けられています．

拡張機能を使用するには次のような準備が必要です．

- インタフェースボード等の準備（一部機能では必要なし）
- 拡張機能制御部のファイルの準備
- BASIC起動時の/E：による拡張機能の使用宣言

**注意**：コンパイラでは，電話制御機能を除いて，インタフェースボード（一部機能で必要なし）等の準備だけが必要です．詳しくは各機能の解説を参照してください．

▶**参照**　拡張機能の使用宣言→「2.3　起動パラメータの指定」

次に各拡張機能について解説します．

## 11.2　GP-IB制御機能

BASICでGP-IB制御機能を使用する場合，次のような準備が必要です．

① GP-IB（IEEE-488）インタフェースボード（PC-9801-29N．以降GP-IBボードと略します）の実装

② BASICの起動時に/E：パラメータで〝GPIB〟（GP-IB機能制御部）を指定する

コンパイラでは①の準備だけでGP-IB制御機能を使用できます．

---

**注意**：GP-IBボードをPC-98XA(XLを除く)で使用する場合は，ROM　KILL状態で使用する必要があります.ROM　KILLの方法についてはGP-IBボードに添付されているユーザーズマニュアルを参照してください.

---

GP-IB制御機能関連の命令の具体的な使用方法に関しては,GP-IBボードに添付されているユーザーズマニュアルおよび「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」を参照してください.

GP-IBボードのユーザーズマニュアルはPC-9800シリーズ$N_{88}$-日本語BASIC(86)の拡張インタプリタの立場で説明されていますが,命令の種類および機能に関しては,$N_{88}$-日本語BASIC(86)(MS-DOS版)は$N_{88}$-日本語BASIC(86)と互換性があります.

## 11.3 BRANCH4670対応ネットワーク制御機能

BASICでBRANCH4670対応ネットワーク制御機能を使用する場合,次のような準備が必要です.

①ネットワークインタフェースセット(PC-9864L)の実装
②MS-DOSに対するBRANCH4670対応拡張ソフトウェアの準備
③本体をネットワークシステムのワークステーションとして立ち上げる
④BASICの起動時に/E：パラメータで〝NETWORK〟(BRANCH4670対応ネットワーク機能制御部)を指定する

コンパイラでは①～③の準備だけでBRANCH4670対応ネットワーク制御機能を使用できます.

BRANCH4670対応ネットワーク制御機能関連の命令の機能に関しては「C&C-NET　BRANCH4670対応MS-DOS　X.X　拡張ソフトウェア　ユーザーズマニュアル」および「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」を参照してください.

# 11.4 MS-NETWORKS 制御機能

BASIC で MS-NETWORKS 制御機能を使用する場合，次のような準備が必要です．

①ネットワークインタフェースセット(PC-9864L)の実装
② MS-DOS(Ver 3.1 以降)と MS-DOS(Ver 3.1 以降)に対応する MS-NETWORKS 対応拡張ソフトフェアの準備
③本体をネットワークシステムのワークステーションとして BASIC 起動前に立ち上げる
④ BASIC の起動時に/E：パラメータで〝MSNET〟(MS-NETWORKS 機能制御部)を指定する．このとき，拡張インタプリタファイル(MSNET．EXE)を MS-NETWORKS のサーバ側で共有する場合，許可スイッチ(r，w，c)はすべて省略するか，またはすべて指定する必要がある

コンパイラでは①～③の準備だけで MS-NETWORKS 制御機能を使用できます．

MS-NETWORKS 制御機能関連の命令の機能に関しては「MS-NETWORKS ユーザーズマニュアル」および「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」を参照してください．なお，BASIC の MS-NETWORKS 制御機能関連の命令と MS-NETWORKS のワークステーションコマンドの対応は次のとおりです．

| BASIC の命令 | ワークステーションコマンド | 機　能 |
|---|---|---|
| CMD CONNECT | Net Use | 装置をサーバの資源に接続する． |
| CMD CONT | Net Continue | 一時的に切り離した装置とサーバの資源を再度接続する． |
| CMD DISCONNECT | Net Use/d | 接続している装置を資源から切り離す． |
| CMD PAUSE | Net Pause | 接続している装置を一時的に，サーバの資源から切り離す． |
| CMD STATE<br>CMD LSTATE | Net Use | ワークステーションの装置とサーバの資源との接続状況を表示／印字する． |

## 11.5 10進演算制御機能

BASICは，通常，数値定数や数値変数(整数型，単精度実数型，倍精度実数型)を，内部の処理では2進数データに変換して扱っています．この場合，入力されたデータ(10進数データ)を内部形式である2進数データに変換する際や，2進数データを出力形式(10進数データ)に変換する際に，変換誤差が生じることがあります．

10進演算機能は，入力データを2進数データに変換せずに10進数データのまま扱うことにより，変換誤差のない演算を行うことができます．

BASICで10進演算機能を使用する場合，次の準備が必要です．

○ BASICの起動時に/E：パラメータで〝DECCAL〞(10進演算機能制御部)を指定する

> コンパイラではそのまま10進演算制御機能を使用できます．

この10進演算機能は，四則演算，指数演算(べき乗)，絶対値演算，整数化演算で使用できます．有効桁は18桁で，扱える範囲は$-9.99999999999999999\times10^{62}$〜$9.99999999999999999\times10^{62}$です．

### ■データ形式

10進演算機能では「10進文字列」「パック10進数」の2つのデータ形式があります．

#### ・10進文字列

10進演算で扱うデータは，あらかじめ10進文字列として定義する必要があります．10進文字列とは，数字，ピリオド，スペース，＋，－および指数表現のためのDまたはEの文字の組み合せで構成される文字列定数または文字列変数です．

例)
```
A$=" 100.56"
B$=" 1.23E-5"
```

10進演算は10進文字列そのものに対して行うことはできません．10進文字列をパック10進数に変換してから演算します．演算結果は再び10進文字列に変換して画面などに出力します．

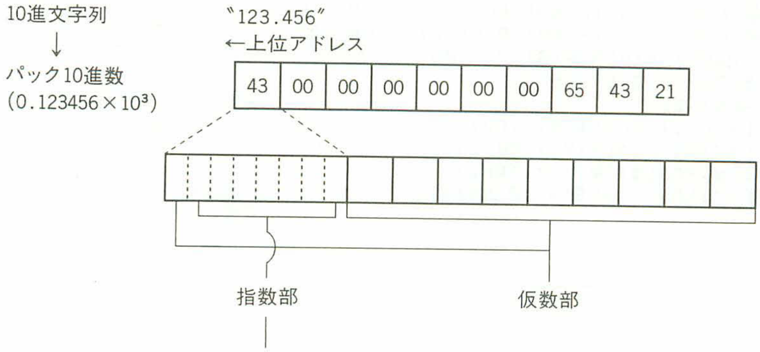
#### ・パック10進数

10進演算は，10進数文字列をパック10進数という内部形式に変換して行います．

パック10進数は10バイト固定長の文字列で，9バイト＋1ビットの仮数部と，7ビットの指数部で構成されています．10進数の各桁は4ビットで表現されています．つまり仮数部9バイ

トで10進数18桁分の有効数字を持っています．

10進文字列からパック10進数への変換とその内部表現の例を次に示します．




指数部の表現形式

| 指数部(16進) | 00 | 01 | … | 3F | 40 | 41 | … | 7E | 7F |
|---|---|---|---|---|---|---|---|---|---|
| 値 | −64 | −63 | … | −1 | 0 | 1 | … | 62 | 63 |

このパック10進数の内部表現をユーザーが意識する必要はありません．10進演算機能では，専用の関数を用いて，データ変換や計算を行うことができます．

| | |
|---|---|
| PCAL$ | 四測演算，指数演算(べき乗)，絶対値演算，整数化演算を行う |
| PCHK | 10 進文字列の妥当性およびパック10進数の相互の大小関係を調べる |
| PCNV | パック 10 進数を倍精度実数に変換する |
| PCNV | 数値，パック 10 進数，10 進文字列の相互の変換，あるいは10進文字例の編集処理を行う |

▶**参照**　各関数の詳細→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

## ■ 10 進演算関数の使用例

10 進演算関数の使用例として，半年複利定期預金の元利合計算出プログラムを考えます．元金(G)，年利(R)，年(Y)とすると元利合計(T)は次の式で表せます．

**元利合計(T)=G ＊(1＋R/2)^(Y/0.5)**

この式を，10 進演算関数を使用して計算するプログラム例を次に示します．ただし，実際の半年複利定期預金の計算は半年単位に計算され，切り捨て処理が施されるため必ずしも計算の結果が一致するわけではありません．10 進演算関数の使用例として参考にしてください．

```
100 CLS
110 DEFSTR A-Z
120 C0.5=PCNV$("0.5"):C1=PCNV$("1") ──────── 定数をパック10進数に変換
130 C2=PCNV$("2"):C100=PCNV$("100")
140 INPUT "元金（円）";G:IF NOT PCHK(G) THEN 140 ──── 入力値の妥当性をチェックし，
150 INPUT "年利（％）";R:IF NOT PCHK(R) THEN 150      不適当な場合は再入力を要求
160 G=PCNV$(G):R=PCAL$(PCNV$(R),/,C100) ──── 変数値(文字列)をパック10進数に変換
170 FOR NEN!=.5 TO 10 STEP .5
180   Y=PCNV$(NEN!,2) ──────────────── 変数値(単精度)をパック10進数に変換
190   TMP1=PCAL$(C1,+,PCAL$(R,/,C2)) ──────── 元利合計の計算
200   TMP2=PCAL$(INT,PCAL$(Y,/,C0.5),1)
210   T=PCAL$(G,*,PCAL$(TMP1,^,TMP2))
220   PRINT "年=";NEN!,"合計=";PCNV$(T,1) ──── 演算結果をパック10進数から文字列に
230 NEXT                                      変換して表示
240 END
```

## 11.6 電話制御機能

BASICで電話制御機能を使用する場合，次の準備が必要です．

①次の機器のいずれかの実装

| モデム-NCU内蔵電話 | PC-TL101 オートホン<br>PC-TL201 オートホン | 1200ボー(半二重)，300ボー(全二重)<br>1200ボー(全二重)，300ボー(全二重) |
|---|---|---|
| モデムボード＋ハンドセット | PC-9863N＋PC-TL901<br>PC-9865＋PC-TL901 | 1200ボー(半二重)，300ボー(全二重)<br>1200ボー(全二重)，300ボー(全二重) |

これらのモデム-NCU内蔵電話機やモデムをコンピュータ本体に接続すれば，BASICの拡張命令を利用して，オートダイアル，自動着信，モデムを利用したデータ通信などを行うことができます．

② BASICの起動の時に/E：パラメータで〝PHONE〞(電話制御機能)を指定する．

コンパイラでは電話制御機能を使用できません．

## ■電話制御のための拡張命令

BASICには次のような拡張命令および関数が用意されています.

| 分類 | 命令・関数 | 機　能 |
|---|---|---|
| 電話機の制御命令 | CMD LINE OPEN<br>CMD LINE CLOSE<br>CMD DIAL<br>CMD STORE DIAL*<br>CMD ON LINE GOSUB<br>CMD LINE ON/OFF/STOP<br>CMD RECEIVE<br>CMD MODE CUT | 電話機とBASICを論理的に接続する<br>電話機とBASICを論理的に切り離す<br>電話をかける<br>電話機に電話番号を記憶させる<br>着信があった時の割り込み先を定義する<br>着信割り込みを制御する<br>着信があった場合の動作を指定する<br>電話を切る |
| 電話機に対する関数 | STATUS LINE<br>STATUS DIAL$*<br>STATUS DIAL*<br>STATUS MODE | 着信があったかどうかを調べる<br>電話機に記憶されている電話番号を調べる<br>電話機に記憶されている電話番号の機能を調べる<br>現在の電話機のモードを調べる |
| 通信関係の命令 | CMD CHANGE DUPLEX<br>CMD BREAK<br>CMD ERROR ON/OFF/STOP<br>CMD ON ERROR GOSUB | 通信方式を切り換える<br>ブレーク信号を送出する<br>通信エラー割り込みを制御する<br>通信エラーがあった時の割り込み先を定義する |
| 通信関係の関数 | STATUS ERROR | 通信エラーの有無あるいはエラーの種類を調べる |

*モデムボードでは使用できません.

▶参照　電話制御命令→「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」

## ■電話制御命令の使用に際して

### ●電話機のスイッチ設定

各モデム-NCU内蔵電話機,モデムボード,ハンドセットのユーザーズマニュアルを参照しながら,ボーレート(転送速度),発信/受信モードなどを設定してください.

### ●電話機とコンピュータ本体の接続

電話機とコンピュータ本体のRS-232Cポートを,電話機に付属しているRS-232Cケーブルで確実に接続してください.RS-232Cポートは標準ポート(第1回線)および拡張ポート(第2回線/第3回線)のいずれも使用できますが,拡張ポートを使用するには,RS-232C拡張インタフェースボードが必要です.

次に,モデム-NCU内蔵電話機の場合,電話機の自動/手動ボタンを押して〝自動〟のランプが点灯するようにセットしてください.手動の場合,オートダイアル,自動着信などの制御は行えません.

### ●ボーレートの指定

・RS-232C 標準ポート（第１回線）使用の場合

第１回線を使用する場合，MS-DOS の SWITCH コマンド，またはセットアップユーティリティ“N88SETUP”を使用して，電話機のボーレートと同じ値(300 ボーあるいは 1200 ボー)に設定します．

▶参照　SWITCH コマンド→「MS-DOS のマニュアル」
N88SETUP →「12.2　セットアップユーティリティ」

・RS-232C 拡張ポート（第２回線あるいは第３回線）を使用する場合

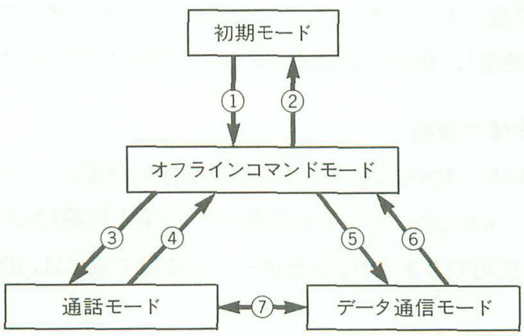
第２回線あるいは第３回線を使用する場合，RS-232C 拡張インタフェースボード上の転送速度設定スイッチの値を電話機のボーレートと同じ値(300 ボーあるいは 1200 ボー)に設定します．

## ■４つのモード

電話制御命令を使い，電話回線を通じてデータ通信などを行う場合には，BASIC，電話機，電話回線を論理的に接続します．この論理的な接続の状態には段階的な４つのレベルがあり，次に説明する４つのモードに分けて考えることができます．

たとえば，電話回線を通じてデータ通信を行う場合には，電話制御命令を使って，「初期モード→オフラインコマンドモード→データ通信モード」へと進んで行きます．

それぞれのモードでは実行できる電話制御命令が異なりますので注意してください．




①：CMD LINE OPENの実行
②：CMD LINE CLOSEの実行
③：CMD DIALの実行，手動ダイヤルまたは手動着信の操作
④：受話器を置く，フックボタンを押す
⑤：CMD DIALの実行，自動着信の操作
⑥：CMD MODE CUTの実行
⑦：通話／データ切り換えボタンによる操作

**●初期モード**

BASICと電話機が論理的に接続されていない状態です．この状態ではオートダイアル，自動着信などの機能を利用することができません．

手動による電話回線の接続あるいはモデム機能のみを利用したデータ通信が可能です．

**●オフラインコマンドモード**

CMD　LINE　OPEN命令を実行すると電話機とBASICが論理的に接続され，オートダイアル，自動着信などの機能が利用できるようになります．このモードを〝オフラインコマンドモード〟といいます．この状態では，電話機と電話回線は切り離されていますが，BASICと電話機との間で命令の受け渡しができます．

CMD　LINE　CLOSE命令を実行すると初期モードにもどります．

**●データ通信モード**

オフラインコマンドモードから，データ通信モード指定のCMD　DIAL命令を実行するかあるいは自動着信が行われると電話回線とBASICがモデムを介して接続された状態になります．これを〝データ通信モード〟といいます．この状態ではコミュニケーションファイルを通じて電話回線上の通信相手とデータの受け渡しをすることができます．このモードのときには電話機のフックランプとデータランプがともに点灯しています．

CMD　MODE　CUT命令を実行するとオフラインコマンドモードにもどります．

また，データ通信モードで受話器を外し，通話／データ切り替えボタンを押すことにより通話モードに移行することができます．

**●通話モード**

オフラインコマンドモードから，通話モード指定のCMD　DIAL命令を実行するかあるいは手動ダイアル，手動着信を行うと，電話回線と電話機とが接続され通話相手と話すことができます．この状態を〝通話モード〟といいます．このモードのときには電話機のフックランプが点灯し，データランプが消灯しています．

このモードではBASICから電話機を制御することはできません．受話器を置くかフックボタンを押すことによって電話を切るとオフラインコマンドモードにもどります．

また，このモードで通話／データ切り替えボタンを押すことによりデータ通信モードに移行することができます．

| モード | 使用できる命令・関数 |
|---|---|
| 初期モード | CMD LINE OPEN 命令<br>STATUS MODE 関数<br>CMD CHANGE DUPLEX 命令<br>コミュニケーションファイルに対する入出力命令<br>(OPEN/CLOSE/PRINT #/INPUT #など) |
| オフライン<br>コマンドモード | CMD LINE CLOSE 命令<br>CMD STORE DIAL 命令*<br>CMD RECEIVE 命令<br>CMD LINE ON/OFF/STOP 命令<br>CMD MODE CUT 命令<br>CMD ON LINE GOSUB 命令<br>STATUS LINE 関数<br>STATUS DIAL$関数*<br>STATUS DIAL 関数*<br>STATUS MODE 関数 |
| データ通信モード | CMD CHANGE DUPLEX 命令<br>CMD BREAK 命令<br>CMD MODE CUT 命令<br>CMD LINE CLOSE 命令<br>STATUS ERROR 関数<br>STATUS MODE 関数<br>コミュニケーションファイルに対する入出力命令<br>(OPEN/CLOSE/PRINT #/INPUT #など) |
| 通話モード | SATUS MODE 関数 |

＊モデムボードでは使用できません．

## ■電話制御命令のプログラム使用例

### ●電話をかけて通話を行う

自動ダイアルで相手に電話をかけるプログラムです．

```
100 ' AUTO DIAL
110 CMD LINE OPEN "COM:"
120 *MAIN
130 INPUT "Telphone number";DL$
140 IF DL$="" THEN CMD LINE CLOSE:END
150 CMD DIAL DL$
160 GOTO *MAIN
```

110行で電話機とBASICを論理的に接続し，初期モードからオフラインコマンドモードに切り換えます．〝COM：″は，コンピュータ本体と電話機がRS-232C標準ポートで接続されていることを意味します．

130行で相手の電話番号を入力します．内線電話から外線にかける場合に電話番号の前に〝0：″などをつける必要がある場合には〝0：XXXX″のように入力します．

電話番号の入力の際に[⏎]キーだけを押すと，140行で電話機とBASICを論理的に切り離

し，初期モードにもどしてプログラムを終了します．

150行でダイアル後，通話モード指定で電話をかけます．ダイアルが終わると次の命令に進みます．この場合再び電話番号入力待ちとなります．

オートダイアルを行うと電話器のフックランプが自動的に点灯し，相手を呼び出し始めます．この間，内蔵スピーカを通して呼び出し音が確認できます．相手が出たら受話器をとって会話します．通話が終わった場合，または話し中あるいは相手が出ないなどの場合には受話器を置くかフックボタンを押して電話を切ってください．

●**電話をかけてデータを送信する**

自動ダイアルで電話をかけファイルの内容を転送するプログラムです．このプログラムは，受信側が次の「自動着信後データを受信する」プログラムを起動させていることを想定しています．

```
100 ' AUTO DIAL & SEND FILE
110 CMD LINE OPEN "COM:".
120 *MAIN
130 INPUT "Telehpone number";DL$
140 IF DL$="" THEN CMD LINE CLOSE:END
150 INPUT "Send file name";FLN$
160 OPEN FLN$ FOR INPUT AS #1
170 CMD DIAL DL$,1
180 OPEN "COM:E71N" AS #2
190 PRINT #2,FLN$
200 WHILE (NOT EOF(1))
210  PRINT #2,"1"
220  LINE INPUT #1,A$
230  PRINT #2,A$
240 WEND
250 PRINT #2,"0"
260 CLOSE
270 CMD MODE CUT
280 GOTO *MAIN
```

まず110行で電話機とBASICを論理的に接続し，初期モードからオフラインコマンドモードに切り換えます．

次に130行で相手の電話番号を入力します．電話番号の入力の際に[⏎]キーだけを押すと，電話機とBASICを論理的に切り離し初期モードにもどしてプログラムを終了します．

150，160行では，ディスク中の転送するファイル名を入力し，そのファイルをオープンします．

170行で電話をかけますが，電話番号の次に〝1″が指定されているのでダイアル後相手のアンサートーンを検出するまで待っています．この命令は送信先の相手が「自動着信後データを受信する」のプログラムを起動して受信待ちの状態になっていなければなりません．そうでない場合には規定時間内にアンサートーンが検出できないのでエラーとなります．

電話回線が接続され，アンサートーンを検出したら電話機に接続されているコミュニケーションファイルをオープンします．この命令を実行することにより以降，モデムを介してPRINT #　，INPUT #などの入出力命令を実行することが可能となります．

なお，1200ボー(半二重)による通信を行う場合，OPEN命令の前に次の命令を実行してください．

```
CMD  CHANGE  DUPLEX  1, 1
```

この命令は通信方式を全二重通信方式から半二重通信方式に切り換える命令です．

190行～250行では，転送するファイル名に続いてファイルのデータを順次相手に送信します．このプログラムでは，始め〝1″を送り，続けてファイルの内容を1行分送る，ということを繰り返しています．ファイルの中身の転送が終わると〝0″を送ります．これはデータの転送が終わったことを知らせる意味を持っています．

データの転送が終わると，260行で電話機に接続しているコミュニケーションファイルと転送したデータファイルをクローズします．

最後に270行で電話を切りオフラインコマンドモードに移行させます．

**●自動着信後データを受信する**

自動着信を行い，受信したデータをディスクファイルに書き出すプログラムです．このプログラムは，送信側が「電話をかけてデータを送信する」のプログラムを起動させて電話をかけることを想定しています．

```
100 ' RECEIVE FILE
110 CMD LINE OPEN "COM:"
120 CMD ON LINE GOSUB *REC
130 CMD LINE ON
140 *LOOP
150 GOTO *LOOP
160 *REC
170 CMD RECEIVE 1
180 IF STATUS MODE<>2 THEN 180
190 OPEN "COM:E71N" AS #1
200 LINE INPUT #1,FLN$
210 PRINT "RECEIVE FILE=";FLN$
220 PRINT "START"
230 OPEN FLN$ FOR OUTPUT AS #2
240 *REC10
250 INPUT #1,A$
260 IF A$<>"1" THEN *REC20
270 LINE INPUT #1,A$
280 PRINT #2,A$
290 GOTO *REC10
300 *REC20
310 CLOSE
320 CMD MODE CUT
330 CMD RECEIVE 0
340 PRINT "END"
350 RETURN
```

まず，110行で電話機とBASICを論理的に接続し，初期モードからオフラインコマンドモードにします．

120，130行では着信があった場合の分岐先を定義し，着信割り込みを許可します．これ以降着信があるとすぐ＊RECルーチンへ分岐します．

その後，着信があるまで待っています．もちろんこの間に別の処理を実行させることも可能です．

170行では着信があるとアンサートーンを返し，その後データ通信モードに移行します．このとき，180行のSTATUS　MODE関数によってデータ通信モードであることを確認してから，190行で電話機に接続されているコミュニケーションファイルをオープンします．以降モデムを介して，PRINT　#，INPUT　#などの入出力命令が実行可能となります．

なお，1200ボー(半二重)で通信を行う場合，OPEN命令の前に次の命令を実行してください．

```
CMD  CHANGE  DUPLEX  1, 1
```

この命令は通信方式を全二重通信方式から半二重通信方式に切り換える命令です．

まず，200行でコミュニケーションファイルから，受信するファイルの名前を受け取ります．その後，ファイル名と受信を始める合図として，〝START〟と表示し，そのファイル名のディスクファイルを出力モードでオープンします(210行～230行)．

240行から290行ではコミュニケーションファイルから順次データを受信し，ファイルに書き込みます．

この場合，〝1〟の次にはファイルの内容が1行送られてくるので，これをファイルに書き込むということを繰り返します．250行で〝1〟以外を受信した場合には電話を切る処理のルーチンにジャンプします．

310行～330行はファイルをクローズし，電話を切り，自動着信を禁止にするサブルーチンです．つまり，再び着信割り込み処理ルーチンに分岐するまではデータ通信モードに切り換わらないようにします．そして〝END〟という受信終了のメッセージを表示して割り込み前の処理(着信待ち)にもどします．

## ■電話制御命令使用時の注意

- PC-TL101，PC-TL102オートホーン，PC-9863N，PC-9865モデムボード以外の電話機を接続した場合の動作は保証できません．
- データ通信モードのときBASICのエラーが発生したり STOP キーを押してプログラムの実行が中断した場合には自動的に電話が切れます．
- データ通信中になんらかの事情により通信できなくなった場合，永久に相手からのデータを待っている状態に陥る可能性があります．無人運転を行う際は時間監視を行い，一定の時間

内にデータが送られてこない場合には電話を切るというようなプログラムを作成してください.

- 電話制御命令を使用しているときにはコミュニケーションポートに対してOUT命令を実行しないでください．動作を保証できません.
- CMD　LINE　OPEN命令でコミュニケーションポートを接続した場合，そのポートに対応するコミュニケーションファイルは必ずデータ通信モード内でOPEN命令とCLOSE命令を実行してください．オープンしたままデータ通信モードを抜け出し，再度データ通信モードとなるとコミュニケーションファイルに対する入出力命令が誤動作する可能性があります.
- 異なった種類の電話機を接続してデータ通信を行う場合，正常動作しなければ受信側のモード設定スイッチはANSモードに設定してください．また，送信側はAUTOあるいはCALLにセッティングします．同一機種を接続する場合，両者ともAUTOモードでかまいません.

## 11.7　サウンド制御機能

サウンド制御機能は，FM音源とSSG音源を利用していろいろな音を発生することのできるものです．BASICでは，このサウンド制御機能を制御することのできる拡張機能が用意されています．ここではサウンド制御機能を使うために準備すべきこと，サウンド制御命令の概要，FM音源とSSG音源の仕組みなどについて解説しています.

BASICでサウンド制御機能を使用する場合，次の準備が必要です.

①サウンドボード(PC-9801-26K)の実装

② BASICの起動時に/E：パラメータで〝SOUND〞（サウンド関連機能制御部）を指定する

コンパイラでは①の準備だけでサウンド制御機能を使用できます.

---

**注意**：PC-9801UV/UX/CVシリーズではサウンドボードを標準実装しています．また，サウンドボードをPC-98XA(XLを除く)で使用する場合は，ROM KILL状態で使用する必要があります．ROM KILLの方法についてはサウンドボードに添付されているユーザーズマニュアルを参照してください.

---

▶**参照**　サウンド制御命令→「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」

## ■サウンド制御機能の特徴

### ●基本機能

・6重和音(FM音源：3声，SSG音源：3声)による音楽演奏が可能.

・8オクターブに渡る音域の発生が可能.

・ビブラート，トレモロなどの効果音制御が可能.

・オーディオ出力端子をオーディオアンプ，ラジオカセットなど外部オーディオ機器に接続することができる．外部オーディオ機器を接続しない場合，本体内のアンプ，スピーカが使用できる.

### ● FM音源

・サウンド発生部にFM(Frequency Moduration)方式音源のLSIを使用.

・FM音源3声に対する多種多様な基本音色の提供により，音色番号を指定するだけで音楽演奏が可能．さらに独自に音色を作ることも可能．種々の効果音も用意されている.

・FM音源3声に対してそれぞれ別の音色を割り当てることができる．たとえばストリング系音色(バイオリン，チェロなど)による三重奏などもできる.

### ● SSG音源

・SSG音源3声に対してエンベロープ形状とエンベロープ周期を設定することにより,音に音量の時間的変化をつけることができる.

・シンセサイザレジスタに直接値を指定することにより音色を変えることができる.

### ● BASICによる音楽演奏

・BASICのサウンド制御命令を使った音楽演奏が可能.

・バックグラウンド演奏が可能．たとえば，グラフィック画面に絵を描きながら音楽演奏を行うというような並列処理が可能.

## ■サウンド制御命令の概要

### ●初期設定　PLAY　ALLOC命令

サウンド制御命令では,演奏のための情報の大部分をPLAY命令で指定します．PLAY命令の情報はいったんサウンドバッファと呼ばれる領域に格納され，この領域から情報が引き出されながら演奏が行われます.

サウンドバッファは，ユーザーが必要な大きさだけ確保する必要があります．サウンドバッファの確保を行うのがPLAY　ALLOC命令です．なお，PLAY　ALLOC命令は音色，音量，テンポ，音の高さなどの初期化も行います.

### ●音楽データの表現　PLAY 命令

音楽演奏に必要な音程，長さ，テンポなどの情報を指示する命令として PLAY 命令があります．PLAY 命令では 6 チャンネルの同時発音が可能です．また，MML(ミュージックマクロランゲージ）という表現を用いて，先に述べた基本的な情報の他に特殊効果や音色の変化など，多くの情報を与えることができます．

### ●音色の設定や変更　VOICE 命令，VOICE　COPY 命令，VOICE　LFO 命令，PLAY 命令，VOICE REG 命令

#### ・FM 音源の音色

69 種 82 個の音色データが用意されています．また，ユーザーが自分の好みに合わせて音色を変化させたり，まったく最初から自分で新しく音色を作り出したりできるように音色操作のための機能も用意されています．VOICE，VOICE　COPY，VOICE　LFO 命令などを使って，音色データの設定，読み出し，一時変更などが可能です．また，PLAY 命令の Y コマンドや VOICE REG 命令で音色を変えることもできます．

#### ・SSG 音源の音色

SSG 音源では VOICE，VOICE　COPY 命令は使用できません．音色の変更は VOICE　LFO 命令や PLAY 命令の Y コマンドや，VIOCE　REG 命令を使用します．

### ●特殊効果の付加

FM 音源，SSG 音源ともに特殊な演奏効果として，音量や音程にゆらぎを与える効果(LFO 効果)が用意されています．FM 音源ではこの効果のためのデータを音色データの一部として設定可能です．また，VOICE　LFO 命令で一時的に変更することも可能です．

SSG 音源のみに使用できる演奏効果として，音量を時間変化させるエンベロープコントロールがあります．これには，PLAY 命令の S コマンド，M コマンドや，VOICE　REG 命令を使用します．

### ●サウンドハードウェア(シンセサイザ LSI)の直接制御　VOICE　REG 命令

サウンドハードウェアに使用されているシンセサイザ LSI(YM-2203)の内部レジスタを操作してより細かな音創りや演奏を可能にする命令として，VOICE　REG 命令が用意されています．FM 音源，SSG 音源の両方に使用できます．

### ●他の BASIC 命令との並行動作　ON　PLAY　GOSUB 命令，PLAY　ON/OFF/STOP 命令

サウンド制御命令を BASIC の他の命令と並行して実行させることができます．たとえば，音楽を演奏させながら絵を描くといったことが可能です．また，長い曲を複数の PLAY 命令に分けて演奏させる時など，音楽が途切れないようになめらかに演奏させることができます．このような場合，ON　PLAY　GOSUB，PLAY　ON/OFF/STOP 命令などを使用します．

## ■サウンドバッファについて

### ●サウンドバッファサイズの決定

PLAY 命令を含んだプログラムを実行すると，PLAY 命令ごとに，指定された MML 文字列がサウンドバッファに格納され，演奏が行われます．

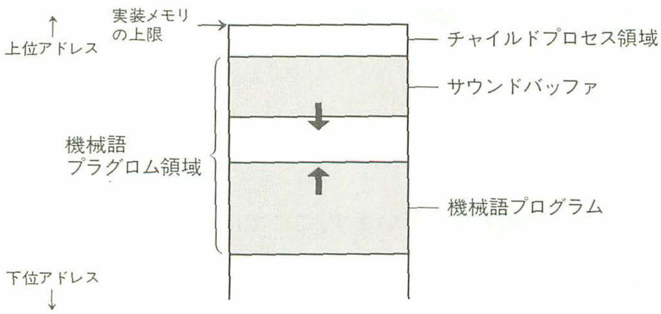
したがって，確保するサウンドバッファのサイズは，メモリに余裕があれば，プログラム中に存在する PLAY 命令のうち最も情報(MML 文字列)の多いものにそのサイズを合わせておくと，演奏がなめらかに行われます．

しかし，メモリに余裕がない場合には，サウンドバッファサイズを小さくして，一つの曲のデータをいくつもの部分(PLAY 命令)に分割して表現する必要があります．

なお，一つの曲を複数の部分に分割した場合，このプログラムをフォアグラウンドで実行すると，途中でとぎれたりしてなめらかに演奏できないことがあります．このような場合には，ON　PLAY(c, n)　GOSUB, STATUS　PLAY などの命令を使って，サウンドバッファに残っている情報の量を監視しながら PLAY 命令を実行するようにしてください．

### ● BASIC のメモリ領域とサウンドバッファの確保される領域

サウンドバッファは機械語プログラム領域の上限から下位アドレスに向かって確保されていきます．




サウンド制御命令と機械語を同時に使用する場合には，サウンドバッファと機械語プログラム領域が重ならないように注意してください．

### ●サウンドバッファの変更

いったん確保したサウンドバッファは新たに PLAY　ALLOC 命令が実行されるまでは有効です．変更後のサイズが CLEAR 命令により確保されている機械語プログラム領域のサイズを超えるような場合は再び CLEAR 命令を実行しなおす必要があります．

なお，PLAY　ALLOC 命令でサウンドバッファのサイズを変更すると，演奏は中断され，初期状態に再設定されます．

▶参照　CLEAR 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

### ●サウンドバッファと他の命令との関連

・PLAY　ALLOC, CLEAR, NEW, LOAD, RUN, ERROR, SYSTEM の各命令

バックグラウンドで演奏が行われているときに，これらの命令が実行されるとサウンドバッファが初期化されますので，演奏は中断されます．

・STOP 命令，[STOP] キー

バックグラウンドでの演奏はいったん停止します．その後，バックグラウンドでの演奏に影響を与えるような命令が実行されず，かつ再実行可能であれば，CONT 命令でバックグラウンドでの演奏も再開されます．また，バックグラウンドでの演奏をいったん停止しているときに，ダイレクトモードでPLAY 命令を実行すると，サウンドバッファに新しいデータが格納されてから，演奏が再開されます．このデータを初期化するためには，PLAY　ALLOC 命令を実行してください．

・サウンドバッファの取扱いに関して

バックグラウンドによる演奏が行われている際に，BLOAD 命令あるいは POKE 命令によりサウンドバッファが乱されますと正しい演奏が行えませんので，アドレス指定には注意が必要です．

・エラーが起きたとき

現在設定中の音色が初期化され演奏が中断されます．

## ■FM 音源について

### ●FM 音源の原理

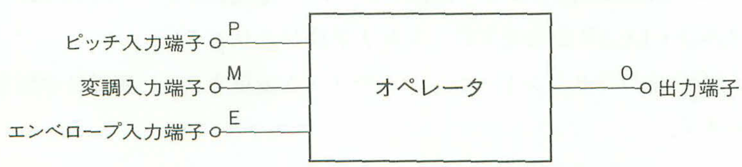
チャンネル 1〜3 は FM 音源となっています．ここでは FM 音源についてその原理を簡単に説明します．

・オペレータとアルゴリズム

FM 音源の基本単位となるものが〝オペレータ〞と呼ばれる回路です．

オペレータは，3 個の入力端子に入れられる信号を電気的に加算したり乗算したりすることによって，波形を合成し，出力端子に出すしくみになっています．

次の図にオペレータを簡略化して示します．



ピッチ入力端子は，波形の周波数(高さ)を決めるための信号を入力するところです．

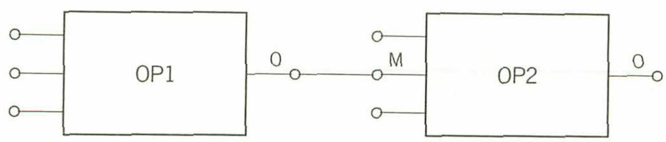
変調入力端子は，波形そのもの(形状)を変化させるための信号を入力するところです．

エンベロープ入力端子は，波形の振幅(強さ)を変化させるための信号を入力するところです．

出力端子は，得られた波形をとりだすところです．

3種類の入力信号をオペレータに入れてやると，出力端子からその入力信号に応じた波形が得られます．これらの入力信号のうち，エンベロープ入力だけはオペレータ外部の別の回路でつくってやります．ピッチ入力と変調入力は，とくに別の回路を使わず，同様な構成をもつ別のオペレータからの出力信号を用います．つまり，オペレータを組み合せて使うわけです．これにより，ひとつのオペレータではつくることのできない複雑な波形を得ることができるようになります．

たとえば，次のような組み合せを考えてみます．




オペレータOP1の出力端子がオペレータOP2の変調入力端子に接続されています．これにより，オペレータOP1の出力信号は，オペレータOP2でつくられる波形そのもの(形状)を変化させるための変調信号となります．このような組み合せの場合，オペレータOP1を〝モジュレータ″，オペレータOP2を〝キャリア″と呼びます．

このシンセサイザLSIには，オペレータが各チャンネルごとに4個ずつ用意されており，これらの4個の組み合せ方をいろいろ変えることができるようになっています．つまり，どのオペレータも，キャリアとしてあるいはモジュレータとして使えるようになっているわけです．この組み合せ方のことを，〝アルゴリズム″と呼び，全部で8種類用意されています．

FM音源の最大の特徴はこのアルゴリズムの自由さにあり，これによって非常に複雑な波形をつくりだすことができるのです．

BASICでは，アルゴリズムをVOICE命令で指定することができます．

---

**注意**：FMとはFrequency　Moduration(周波数変調)のことです．FM音源は，オペレータ中で変調を行う際にこの方式を用いることから，こう呼ばれます．

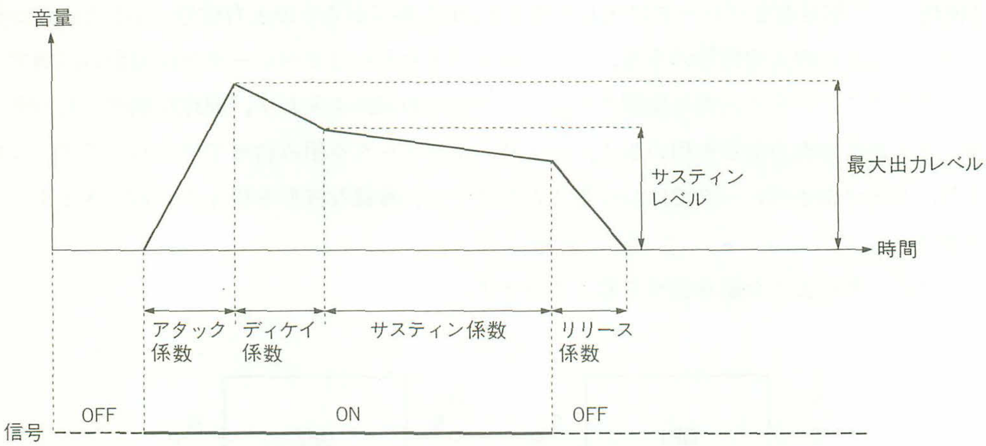
---

**・エンベロープジェネレータ**

前の項で，〝エンベロープ入力だけは別の回路を使って信号をつくる″と説明しましたが，これを行う回路がエンベロープジェネレータ(EG)で，シンセサイザLSIに内蔵されています．

エンベロープジェネレータは音量の時間的変化を表す次のようなパラメータを持っており，音にさまざまな表情を付け加えることができます．

BASICで具体的にこれらのパラメータを指定するには，VOICE命令を使用します．




**アタック係数(Attack　Rate：AR)**

発音してからEGの出力レベルが最大になるまでの時間を定めます．

最短：0，最長：31

**ディケイ係数(Decay　Rate：DR)**

EGの出力レベルが最大からサスティンレベルになるまでの時間を定めます．

最短：0，最長：31

**サスティン係数(Sustain　Rate：SR)**

サスティンレベルからEGの出力レベルが0になるまでの時間を定めます．

最短：0，最長：31

**リリース係数(Release　Rate：RR)**

発音が終わってからEGの出力レベルが0になるまでの時間を定めます．

最短：0，最長：15

**最大出力レベル(Total　Level：TL)**

EGの最大出力レベルを定めます．

最小：0，最大：127

**サスティンレベル(Sustain　Level：SL)**

Decay状態からSustain状態に移るときのレベルを定めます．

最小：0，最大：15

### ●音色パラメータの設定

サウンド制御命令には，FM 音源での音創りを容易に行うための命令が用意されています．これらの命令を使って，〝音色パラメータ〞(音色を決定する要素)を値として設定することにより音色を変更したり創り出したりすることができます．

| 音色を設定する命令 | 機　能 |
|---|---|
| PLAY 命令の@X コマンド | 各チャンネルに音色を設定する |
| PLAY 命令の Z コマンド，VOICE　LFO 命令 | 音に LFO 効果を与える |
| VOICE 命令 | 事前に配列に入れておいた値を音色パラメータとして設定する．アルゴリズムやエンベロープの各係数など，各種の設定が可能 |

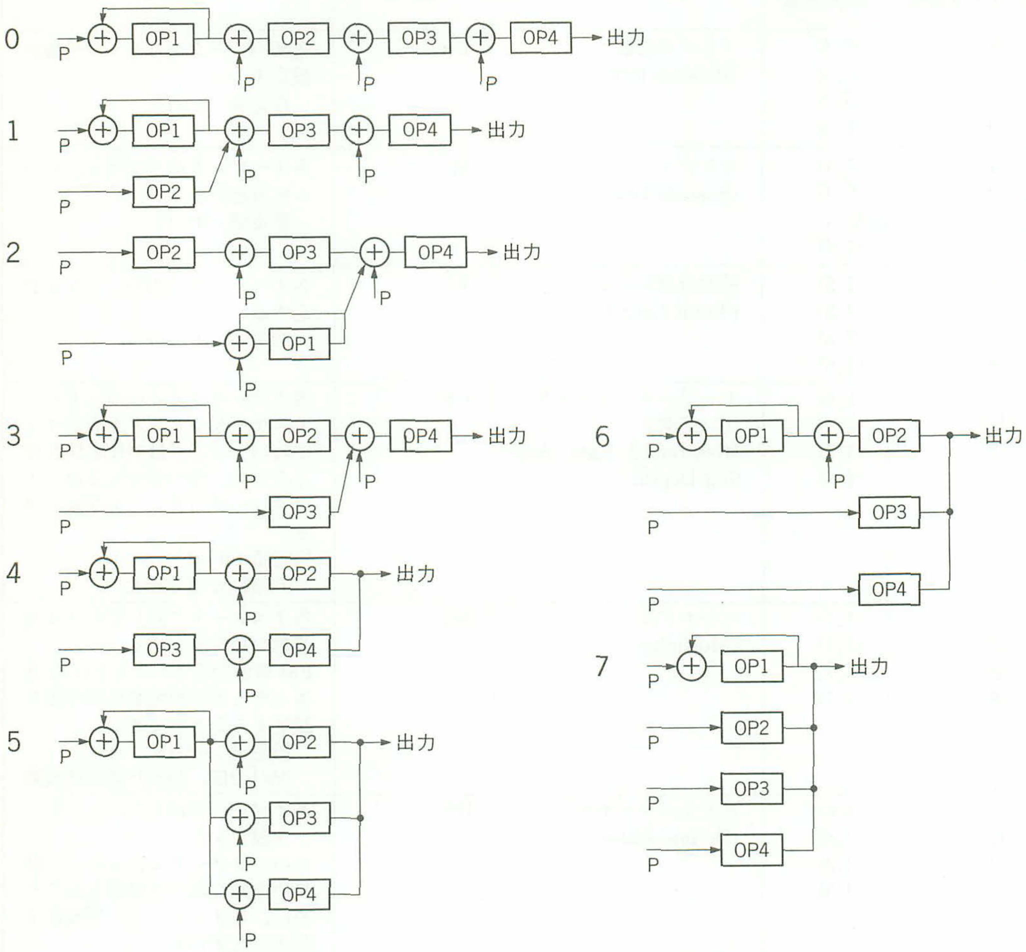
次に音色パラメータのおのおのについて表で解説します．

<table>
<tr><th>音色パラメータ番号</th><th>音色用配列要素</th><th>パラメータ名</th><th>略称名</th><th>解　説</th></tr>
<tr><td>0</td><td>(0,0)</td><td>フィードバック／アルゴリズム<br>(Feed Back／Algorith-m)</td><td>FB／ALG</td><td>6ビットからなる値<br>7　6　5　4　3　2　1　0<br>× × $F_2$ $F_1$ $F_0$ $A_2$ $A_1$ $A_0$<br>フィードバック　アルゴリズム<br>・フィードバック<br>$F_2$～$F_0$の3ビットを使用する<br>第1オペレータは自分の出力を変調入力とするが，その変調度を決定する<br><table><tr><th>$F_2$～$F_0$</th><th>変調度</th></tr><tr><td>0</td><td>OFF</td></tr><tr><td>1</td><td>$\pi/16$</td></tr><tr><td>2</td><td>$\pi/8$</td></tr><tr><td>3</td><td>$\pi/4$</td></tr><tr><td>4</td><td>$\pi/2$</td></tr><tr><td>5</td><td>$\pi$</td></tr><tr><td>6</td><td>$2\pi$</td></tr><tr><td>7</td><td>$4\pi$</td></tr></table>・アルゴリズム<br>$A_2$～$A_0$の3ビットを使用する<br>4つのオペレータのアルゴリズムを決定する<br>$A_2$～$A_0$の値とアルゴリズムの関係は「アルゴリズムの種類」の図参照</td></tr>
</table>

| 音色パラメータ番号 | 音色用配列要素 | パラメータ名 | 略称名 | 解　説 |
|---|---|---|---|---|
| 5 | (0,1) | オペレータマスク<br>(Operater Mask) | OPMSK | 4ビットからなる値<br>指定値：0〜15<br>7 6 5 4 3 2 1 0<br>× × × × $O_4$ $O_3$ $O_2$ $O_1$<br>オペレータ ON/OFF<br>$O_4$〜$O_1$の4ビットを使用する．各オペレータの使用／非使用を決定する<br>$O_4$〜$O_1$の各ビットが"1"のとき使用，"0"のとき不使用となる |
| 10 | (0,2) | LFO変調波形<br>(LFO Wave Form) | WF | LFOの変調波形を決定する<br>指定値：0〜5<br>指定値　波形<br>0　ノコギリ波<br>1　矩形波<br>2　三角波<br>3　サンプル&ホールド<br>4　ノコギリ波ショット<br>5　三角波ワンショット |
| 15 | (0,3) | LFO Sync ディレイ<br>(LFO Sync Delay) | SYNC | LFOのSyncディレイタイムを決定する<br>指定値：0〜255 |
| 20 | (0,4) | LFO速さ<br>(LFO Speed) | SPEED | LFOの速さを設定する<br>指定値：0〜16383 |
| 25 | (0,5) | LFOピッチ変調深さ<br>(LFO Pitch Depth) | PD | LFOピッチ変調のかかる深さを細かく調整する<br>指定値：−127〜127 |
| 30 | (0,6) | LFO振幅変調の深さ<br>(LFO Amp Depth) | AD | LFO振幅変調のかかる深さを細かく調整する<br>指定値：−127〜127 |
| 35 | (0,7) | LFOピッチ変調深さ<br>(LFO Pitch Sense) | PS | LFOピッチ変調のかかる深さを大まかに調整する<br>指定値：0〜15 |
| 40<br>45 | (0,8)<br>(0,9) | 未使用 | | |
| 1<br>2<br>3<br>4 | (1,0)<br>(2,0)<br>(3,0)<br>(4,0) | アタック係数<br>(Attack Rate) | AR | 各オペレータのアタック係数を設定する<br>指定値：0〜31 |
| 6<br>7<br>8<br>9 | (1,1)<br>(2,1)<br>(3,1)<br>(4,1) | ディケイ係数<br>(Decay Rate) | DR | 各オペレータのディケイ係数を設定する<br>指定値：0〜31 |
| 11<br>12<br>13<br>14 | (1,2)<br>(2,2)<br>(3,2)<br>(4,2) | サスティン係数<br>(Sustain Rate) | SR | 各オペレータのサスティン係数を設定する<br>指定値：0〜31 |

| 音色パラメータ番号 | 音色用配列要素 | パラメータ名 | 略称名 | 解　説 |
|---|---|---|---|---|
| 16<br>17<br>18<br>19 | (1,3)<br>(2,3)<br>(3,3)<br>(4,3) | リリース係数<br>(Release Rate) | RR | 各オペレータのリリース係数を設定する<br>指定値：0～15 |
| 21<br>22<br>23<br>24 | (1,4)<br>(2,4)<br>(3,4)<br>(4,4) | サスティンレベル<br>(Sustain Level) | SL | 各オペレータのサスティンレベルを設定する<br>指定値：0～15 |
| 26<br>27<br>28<br>29 | (1,5)<br>(2,5)<br>(3,5)<br>(4,5) | 最大出力レベル<br>(Total Level) | TL | 各オペレータの出力レベルを設定する<br>指定値：0～127 |
| 31<br>32<br>33<br>34 | (1,6)<br>(2,6)<br>(3,6)<br>(4,6) | キーボードレイトスケーリング深さ<br>(Key-board Rate Scaling Depth) | KS | 各オペレータのキーボードレイトスケーリング深さを設定する<br>このパラメータは音に表情を与えるため，高い音になるほどエンベロープの各レイトを短くする<br>指定値：0～3<br>0で最小，3で最大 |
| 36<br>37<br>38<br>39 | (1,7)<br>(2,7)<br>(3,7)<br>(4,7) | マルチプル<br>(Multiple) | ML | 各オペレータに対して与える周波数の比を設定する<br>FM音源の各オペレータには，基本となる周波数整数比倍の値を設定することができる<br>指定値：0～15<br>0が1/2倍，15が15倍の周波数 |
| 41<br>42<br>43<br>44 | (1,8)<br>(2,8)<br>(3,8)<br>(4,8) | デチューンレイト<br>(Detune Rate) | DT | 各オペレータのデチューンレイトを設定する<br>このパラメータは，コーラス効果や微妙なゆらぎ効果を出すために，各オペレータの周波数をわずかにずらす<br>指定値：−4～3 |
| 46<br>47<br>48<br>49 | (1,9)<br>(2,9)<br>(3,9)<br>(4,9) | LFO振幅変調深さ<br>(LFO Amp Sense) | AS | 各オペレータのLFO振幅変調深さを，大まかに設定する<br>指定値：0～15 |

**アルゴリズムの種類**




0～7は音色パラメータ番号0の$A_2$～$A_0$の値を示す．Pはピッチ入力を示す．

音色パラメータは，VOICE　LOF命令では有効範囲のチェックが行われますが，VOICE命令やPLAY命令のZコマンドではチェックされません．そのために思ったような音が出ないことがありますので注意してください．

音色パラメータはシンセサイザLSIの内部レジスタと密接に関係しており，LFO関連のパラメータ以外は対応するレジスタが必ず1つは存在します．たとえば，パラメータML(マルチプル)とDT(デチューンレイト)は，内部レジスタの&H30～&H3E番の下位4ビットと上位3ビットに対応します．その対応は「FM/SSG音源内部レジスタマップ」の表に示してあります．VOICE REG命令やPLAY命令のYコマンドで直接内部レジスタを操作する場合には，この表を参考にしてください．

---

**注意**：OUT命令によりシンセサイザLSIレジスタを操作した場合，サウンド制御命令の動作は保証できません．

VOICE　REG命令およびPLAY命令のYコマンドにより＆H21～＆H27番，＆H2D～＆H2F番のレジスタを操作しないでください．

もし，これらのレジスタに値を書いたり読んだりした場合，サウンド制御命令の動作は保証されません．

LFO関係のパラメータのうちピッチ変調に関するものだけは，SSG音源に対しても有効です．

---

### ● FM音源チャンネルCH3のモード

チャンネルのうちFM音源のCH3には以下のような3つのモードがあります．モードの選択はPLAY命令で行います．各オペレータの周波数の設定はVOICE　REG命令を使って直接レジスタに指定します．

**・楽音モード（モード1）**

通常の音楽演奏に適したモードです．

発生周波数をレジスタで直接指定する場合，＆HA2，＆HA6に指定します．

**・効果音モード（モード2）**

効果音の発生に適するように各オペレータは独立して発生周波数を設定できます．

第1オペレータ：＆HA8，＆HAC

第2オペレータ：＆HA9，＆HAD

第3オペレータ：＆HAA，＆HAE

第4オペレータ：＆HA2，＆HA6

**・CSMモード（モード3）**

発生周波数の設定は効果音モードと同様です．ただし，このモードの時には内部タイマレジスタAのカウンタがオーバーフローしたときに音を発生します．

### ● FM音源チャンネルの発生周波数の求め方

FM音源のCH1～3に与える周波数データ（BlockおよびF-number）は次の式で計算します．発生する周波数と，その周波数のオクターブをもとにして，Block値とF-number値を計算します．

Block＝〈オクターブデータ〉

F-number＝(〈発生する周波数(Hz)〉 ＊ 2^20／55556)／2^(Block－1)

例）PLAY 命令における O4A の音(440Hz)の場合．

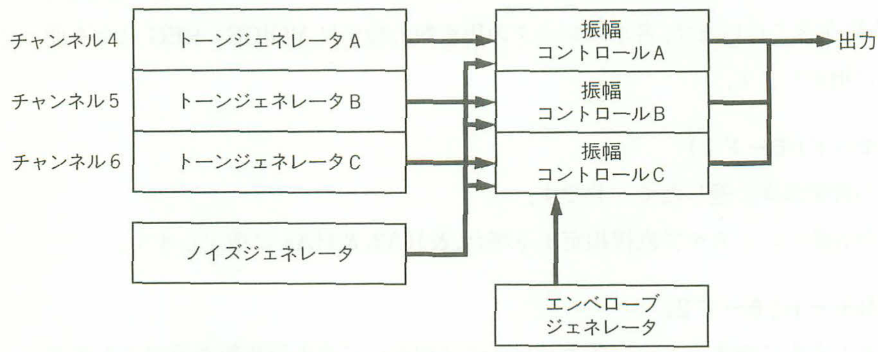
Block＝4

F-number＝(440 ＊ 2^20／55556)／2^(4−1)＝1038

## ■ SSG 音源について

チャンネル 4〜6 は SSG 音源となっています．

SSG 音源部の構成は次のようになっています．レジスタの値を設定することにより音質が変化します．VOICE　REG 命令，PLAY 命令の Y コマンドで直接設定を行うことができます．「FM/SSG 音源内部レジスタマップ」を参照してください．なお，エンベロープ形状とエンベロープ周期は PLAY 命令の S コマンド，M コマンドで設定することもできます．




・トーンジェネレータ：[レジスタ＆H00〜＆H05]

デューティ比 1：1 の方形波を発生する発振器です．

各チャンネルの発振周波数は次の式により求められます．

Fnote＝667 ＊ 1000／(32 ＊ To)

Fnote：発振周波数(Hz)

To：トーンジェネレータ設定値(0〜4095)

・ノイズジェネレータ：[レジスタ＆H06]

類似ランダムノイズを発生します．レジスタ＆H06 に設定される値によりノイズ周波数を可変できます．

Fnois＝667 ＊ 1000／(32 ＊ Tn)

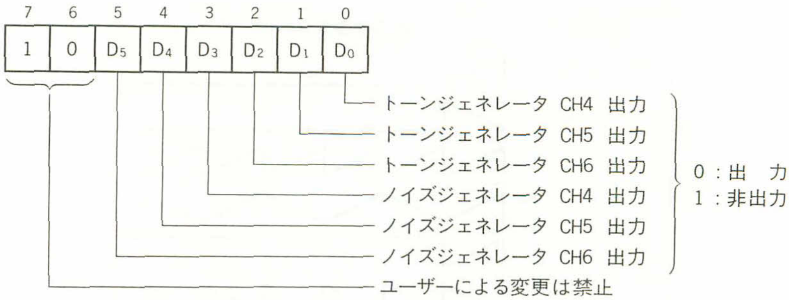
Fnois：ノイズ周波数(乱数系列発生周期)

Tn：ノイズジェネレータ設定値(0〜31)

・ミキサコントロール：[レジスタ＆H07]

トーンジェネレータとノイズジェネレータを出力チャンネルに出すか否かを設定します．

このレジスタは次のようなビット構成になっています．



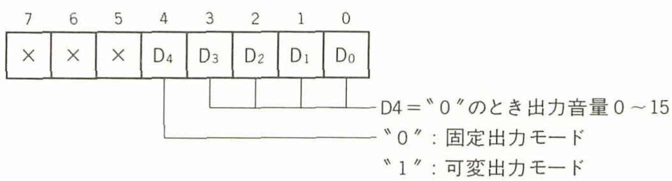

---

**注意**：レジスタ＆H07のビット6，7は〝10″以外を設定してはいけません，〝10″以外の設定を行った場合，動作は保証されません．

---

・振幅コントロール[レジスタ＆H08～＆H0A]

エンベロープジェネレータによる可変出力モード使用の選択，および，固定出力モード時の出力音量を制御します．SSG音源に対するPLAY命令のVコマンドはこのレジスタへの設定を行います．




エンベロープジェネレータコントロール[レジスタ＆H0B～＆H0D]

変化のあるエンベロープを作り出すために2つの値を設定します．

1つはレジスタ＆H0B，＆H0Cに設定するエンベロープ周期で，PLAY命令のMxコマンドに対応します．

エンベロープ周波数は次の式で求められます．

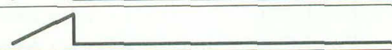
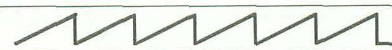
Fenv＝667 ＊ 1000／(512 ＊ Te)

Fenv：エンベロープ周波数(Hz)

Te：エンベロープ周期設定値(0～65535)

もう１つは，レジスタ&H0Dに設定するエンベロープ形状でこれはPLAY命令のSxコマンドに対応します．

レジスタ&H0Dへの設定値とエンベロープの形状の関係は次のとおりです．

| 設定値 | 対応するエンベロープ |
| --- | --- |
| 0～3，9 |  |
| 4～7，15 |  |
| 8 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |

---

**注意**：タイマレジスタ(レジスタ番号&H24～&H27)はサウンド制御命令が内部で使用します．
　　　書き込みを行った場合は動作の保証はありません．

---

## ■ FM/SSG音源内部レジスタマップ

| レジスタ番号＊ | レジスタのビット構成 $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | 対応チャンネル | 説　明 | 備考 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| &H21 | TEST | | | | | | | | × | テスト用レジスタ | ユーザーアクセス禁止 |
| &H24 | TIMER-A(I) | | | | | | | | × | タイマAレジスタ(上位) | ユーザーアクセス禁止 |
| &H25 | ―― | | | | | | TIMER-A(II) | | × | タイマAレジスタ(下位) | ユーザーアクセス禁止 |
| &H26 | TIMER-B | | | | | | | | × | タイマBレジスタ | ユーザーアクセス禁止 |
| &H27 | MODE | | RESET B A | | ENABL B A | | LOAD B A | | × (CH3) | CH3モードおよびタイマ制御 | ユーザーアクセス禁止 |
| &H28 | OPERATOR 4 | 3 | 2 | 1 | ―― | | CH | | 1～3 | 発音／非発音 | |
| &H30<br>&H31<br>&H32 | ― | $DT_1$ | | | $ML_1$ | | | | 1<br>2<br>3 | デチューン/マルチプライ(オペレータ1) | |
| &H34<br>&H35<br>&H36 | ― | $DT_3$ | | | $ML_3$ | | | | 1<br>2<br>3 | デチューン/マルチプライ(オペレータ3) | |
| &H38<br>&H39<br>&H3A | ― | $DT_2$ | | | $ML_2$ | | | | 1<br>2<br>3 | デチューン/マルチプライ(オペレータ2) | |
| &H3C<br>&H3D<br>&H3E | ― | $DT_4$ | | | $ML_4$ | | | | 1<br>2<br>3 | デチューン/マルチプライ(オペレータ4) | |
| &H40<br>&H41<br>&H42 | ― | $TL_1$ | | | | | | | 1<br>2<br>3 | 最大出力レベル(オペレータ1) | ＊＊ |
| &H44<br>&H45<br>&H46 | ― | $TL_3$ | | | | | | | 1<br>2<br>3 | 最大出力レベル(オペレータ3) | ＊＊ |
| &H48<br>&H49<br>&H4A | ― | $TL_2$ | | | | | | | 1<br>2<br>3 | 最大出力レベル(オペレータ2) | ＊＊ |
| &H4C<br>&H4D<br>&H4E | ― | $TL_4$ | | | | | | | 1<br>2<br>3 | 最大出力レベル(オペレータ4) | ＊＊ |
| &H50<br>&H51<br>&H52 | $KS_1$ | | ― | $AR_1$ | | | | | 1<br>2<br>3 | キーボードレートスケーリング深さ／アタック係数(オペレータ1) | ＊＊ |
| &H54<br>&H55<br>&H56 | $KS_3$ | | ― | $AR_3$ | | | | | 1<br>2<br>3 | キーボードレートスケーリング深さ／アタック係数(オペレータ3) | ＊＊ |
| &H58<br>&H59<br>&H5A | $KS_2$ | | ― | $AR_2$ | | | | | 1<br>2<br>3 | キーボードレートスケーリング深さ／アタック係数(オペレータ2) | ＊＊ |
| &H5C<br>&H5D<br>&H5E | $KS_4$ | | ― | $AR_4$ | | | | | 1<br>2<br>3 | キーボードレートスケーリング深さ／アタック係数(オペレータ4) | ＊＊ |

| レジスタ番号＊ | レジスタのビット構成 | | | | | | | | 対応チャンネル | 説　明 | 備考 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | | | |
| &H60<br>&H61<br>&H62 | — | | | $DR_1$ | | | | | 1<br>2<br>3 | ディケイ係数<br>(オペレータ1) | ＊＊ |
| &H64<br>&H65<br>&H66 | — | | | $DR_3$ | | | | | 1<br>2<br>3 | ディケイ係数<br>(オペレータ3) | ＊＊ |
| &H68<br>&H69<br>&H6A | — | | | $DR_2$ | | | | | 1<br>2<br>3 | ディケイ係数<br>(オペレータ2) | ＊＊ |
| &H6C<br>&H6D<br>&H6E | — | | | $DR_4$ | | | | | 1<br>2<br>3 | ディケイ係数<br>(オペレータ4) | ＊＊ |
| &H70<br>&H71<br>&H72 | — | | | $SR_1$ | | | | | 1<br>2<br>3 | サスティン係数<br>(オペレータ1) | ＊＊ |
| &H74<br>&H75<br>&H76 | — | | | $SR_3$ | | | | | 1<br>2<br>3 | サスティン係数<br>(オペレータ3) | ＊＊ |
| &H78<br>&H79<br>&H7A | — | | | $SR_2$ | | | | | 1<br>2<br>3 | サスティン係数<br>(オペレータ2) | ＊＊ |
| &H7C<br>&H7D<br>&H7E | — | | | $SR_4$ | | | | | 1<br>2<br>3 | サスティン係数<br>(オペレータ4) | ＊＊ |
| &H80<br>&H81<br>&H82 | $SL_1$ | | | | $RR_1$ | | | | 1<br>2<br>3 | サスティンレベル／リリース係数<br>(オペレータ1) | ＊＊ |
| &H84<br>&H85<br>&H86 | $SL_3$ | | | | $RR_3$ | | | | 1<br>2<br>3 | サスティンレベル／リリース係数<br>(オペレータ3) | ＊＊ |
| &H88<br>&H89<br>&H8A | $SL_2$ | | | | $RR_2$ | | | | 1<br>2<br>3 | サスティンレベル／リリース係数<br>(オペレータ2) | ＊＊ |
| &H8C<br>&H8D<br>&H8E | $SL_4$ | | | | $RR_4$ | | | | 1<br>2<br>3 | サスティンレベル／リリース係数<br>(オペレータ4) | ＊＊ |
| &HA0<br>&HA1<br>&HA2 | F-number 1 | | | | | | | | 1<br>2<br>3 | F-number 下位 | ＊＊＊ |
| &HA4<br>&HA5<br>&HA6 | — | | Block | | | F-number2 | | | 1<br>2<br>3 | Block/F-number 上位 | ＊＊＊ |
| &HA8<br>&HA9<br>&HAA | CH3<br>F-number 1 | | | | | | | | 3 | F-number 下位(オペレータ1)<br>F-number 下位(オペレータ2)<br>F-number 下位(オペレータ3) | MODE<br>2,3の<br>ときのみ |
| &HAC<br>&HAD<br>&HAE | — | | CH3<br>Block | | | CH3<br>F-number2 | | | 3 | Block/F-number上位(オペレータ1)<br>Block/F-number上位(オペレータ2)<br>Block/F-number上位(オペレータ3) | MODE<br>2,3の<br>ときのみ |
| &HB0<br>&HB1<br>&HB2 | — | | FB | | | ALG | | | 1<br>2<br>3 | ファードバック／アルゴリズム | |

| レジスタ番号＊ | レジスタのビット構成 | | | | | | | | 対応チャンネル | 説　明 | 備考 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | | | |
| &H00 | TUNE11 | | | | | | | | 4 | 周波数レジスタ下位 | |
| &H01 | ―― | | | | TUNE12 | | | | 4 | 周波数レジスタ上位 | |
| &H02 | TUNE21 | | | | | | | | 5 | 周波数レジスタ下位 | |
| &H03 | ―― | | | | TUNE22 | | | | 5 | 周波数レジスタ上位 | |
| &H04 | TUNE31 | | | | | | | | 6 | 周波数レジスタ下位 | |
| &H05 | ―― | | | | TUNE32 | | | | 6 | 周波数レジスタ上位 | |
| &H06 | ―― | | | NOISE | | | | | 4〜6 | ランダムノイズ周波数 | |
| &H07 | 1 | 0 | NOISE MIX | | | TONE MIX | | | 4〜6 | ノイズ／楽音ミキサーコントロール | 上位2bit 変更禁止 |
| &H08<br>&H09<br>&H0A | ―― | | | F/V | LEVEL | | | | 4<br>5<br>6 | 固定量／可変音量コントロール<br>出力音量 | |
| &H0B | ENVELOP PER₁ | | | | | | | | 4〜6 | エンベロープ周期下位 | |
| &H0C | ENVELOP PER₂ | | | | | | | | 4〜6 | エンベロープ周期上位 | |
| &H0D | ―― | | | | SHAPE | | | | 4〜6 | エンベロープ形状 | |

＊ レジスタ番号&H00〜&H0DはSSG用，&H21〜&HB2はFM用となっています．また，内部レジスタのうち未定義のものやユーザーアクセスが禁止されているものをアクセスした場合，その動作は保証されません．

＊＊ 内部レジスタの設定値のうち，TL，AR，DR，SR，SL，RRは，VOICE命令における音色パラメータの設定値とは大小／長短関係が逆になっています．

例）TLの場合

| | 大↔小 |
|---|---|
| 内部レジスタの設定値 | 0 〜127 |
| 音色パラメータの設定値 | 127〜 0 |

＊＊＊ MODE 2,3のときは，&HA2がCH3のオペレータ4のF-number 1(下位)となり，&HA6がCH3のオペレータ4のBlock/F-number 2(上位)となります．

## ■ミュージックジェネレータボード(PC-9801-14)用プログラムの移植

サウンド制御命令はミュージックジェネレータボードで使用するミュージック制御命令と高い互換性があります．したがってミュージックジェネレータ用のプログラムをサウンド制御命令用に移植することも，PLAY命令のMML(ミュージックマクロランゲージ)単位では大きな変更なしに行うことができます．移植の際に注意すべき点は次のとおりです．

サウンド制御命令では6声までの出力ですが，ミュージック制御命令では8声までが使えます．移植の際は2チャンネル分を削ってください．

PLAY命令では次のような変更が必要です．

### ●Sコマンド

・**ミュージック制御命令**

**チャンネル**：全チャンネルで使用可能

**書　　式**：S<アタック係数>，<ディケイ係数>，<サスティンレベル>，<リリース係数>，<サスティン係数>

・**サウンド制御命令**

**チャンネル**：チャンネル4〜6でのみ有効

**書　　式**：S <エンベロープ形状>

チャンネル1〜3に対するエンベロープは音色を切り換えるかユーザーが音色を作成するかしてください．なお，音色パラメータのエンベロープパラメータの数値とミュージック制御命令のSコマンドの数値には互換性はありません．

### ●Oコマンド

・**ミュージック制御命令**

O4〜O7まで可能

・**サウンド制御命令**

O1〜O8(O9)まで可能

### ●Kコマンド

・**ミュージック制御命令**

K1〜K49まで可能．O4Cに対応するものはK1

・**サウンド制御命令**

K0〜K96まで可能．O4Cに対応するものはK36

### ●Tコマンド

・**ミュージック制御命令**

各チャンネルごとにテンポを設定可能．

・サウンド制御命令

全チャンネルを通じて1つ.

PLAY 命令の文字列では，記述されている文字列の最後のテンポ指定が有効.

たとえば次の場合，

PLAY "T96CDE", "T120ABC", "T80FGA"

チャンネル3の文字列中にある〝T80〞が有効となる.

●その他

ミュージック制御命令は1小節を内部で128分割していますが，サウンド制御命令は1小節を192分割しますので，〝L12〞や〝L24〞などの音長も正確に表現できます.

なお，音長が255を超えた場合は，その値を256で割った余りが有効な値となります. 長い音符を表現する際には注意が必要です.

例1) 2分音符の場合.

2分音符=192/2=96

96の長さ，つまり半小節分が発音される.

例2) 符点全音符の場合.

符点全音符=192/2+96=288 → 288÷256=1…32

実際には288を256で割った余りの長さ=32(2拍3連符の1音分)が発音されるため，符点全音符とはならない.

## ■サウンド制御命令を使ったプログラム例

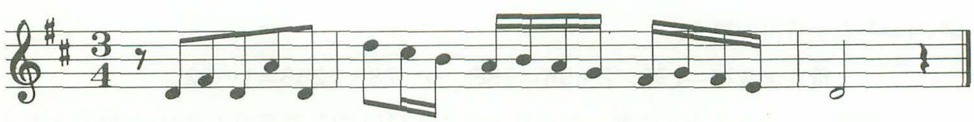
●演奏プログラム

〝バッハのインベンション〞を次に示します.

```
1000 '******************************
1010 '
1020 '            BACH
1030 '                 Invention
1040 '
1050 '******************************
1060 CLEAR &H100
1070 PLAY ALLOC 500,500,500
1080 PLAY "@46@V95MFZ0,3","@46@V95MFZ0,3","@46@V95MFZ0,3"
1090 '
1100 PLAY "T70"
1110 PLAY "L16RO5DC+DO4A8O5F8.EDEO4A8O5G8",,"O4L8DEFDABO5C+O4A"
1120 PLAY "L16RFEDA4RG+F+G+G+8.A","L16RO4AG+AE8O5C8.O4BABE8O5D8","L16O5D4RCO4B
AO5E4O4E4"
1130 PLAY "O5L16A4RDC+DG4RC+O4BO5C+","O3A4","L16RCO4BO5CF4RO4BABO5E4"
1140 PLAY "O4RDC+DO3A8O4F8.EDEO3A8O4G8","O5F4.A8G4RFEF","L16R8O4A8O5DEDCO4B8.O
5C+C+4"
1150 PLAY "RFEDB-4REDCA4","D4RB-AB-E4RAGA","RDC+DO4G4RO5CO4B-O5CO4F4"
```

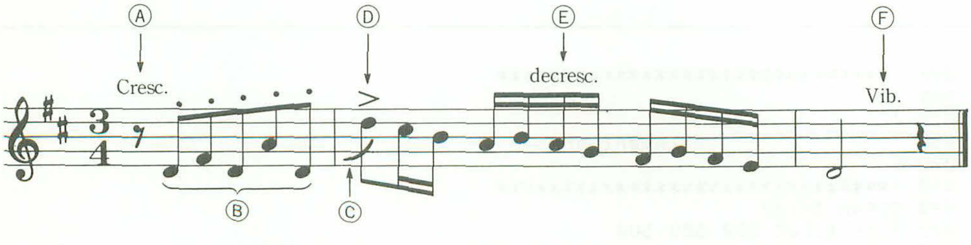
```
1160 PLAY "RDCDO3G8O4F8.EDEC8B-8","D4RDEFG4RFED","RB-AO5CO4B-4RO5CO4B-AB-AGF"
1170 PLAY "RAGAL8FO5E-DCO4B-A","C2CO4B-AB-GO5DCD","E8B-8.AGAF2"
1180 PLAY "B-GO5CO4CL16RFEFC8A8","O4GB-AB-8O5CO4B-O5CO4A4R8O5F8","R4O4E4O3L8FG
AF"
1190 PLAY "RGFGC8B-8.O5FEFC8A8","RE8.R8G8.O4A8.R8O5F8","O4CDECFGAF"
1200 PLAY "RGFGC8B-8.O5CO4B-O5CO4F8O5F8","RE8.R8G8.R4..","O5CDECF4L16REDC"
1210 PLAY "RO4B-AB-E8O5E8.O4AGAD8O5D8","RDC+DG4RFEGF4","O4B-4RAB-GO5C+4DO4B-AB
-"
1220 PLAY "R8C8.O4BABG+8A8.G+F+G","REDEO4A8O5D8.CO4BO5CO4F+8B8","L8G+AF+FE4D+D
"
1230 PLAY "L8AD+EO3EL16O4RAGAE8O5C8","RAG+A8GF+GO3A4R4","L8C+CO3B4ABO4CO3A"
1240 PLAY "RO4BABE8O5D8.O4ED+EO3A8O4G8","R2O5RCO4BO5CE-4","O4EF+G+EA4L16RGF+G"
1250 PLAY "RF+EFD8O5C8.GF+GD8B-8","RD8.R8F+8G2","D8C8O3B-8A8GO4B-AB-O5CDCD"
1260 PLAY "RAGAD8O5C8.DCDO4G8O5F8","RF+B-AF4G","E-4RDE-CDO4B-AB-O5CO4B-AG"
1270 PLAY "REDEC8B-8.O4FEFC8A8","R2RAGB-AGFE","O5C8D8E8C8O4F4R8F8"
1280 PLAY "RGFGO3B-8O4G8.FEFO3A8O4F8","D4RFEDC+4RO4AO5DC","L8B-AGB-AGFD"
1290 PLAY "FEDFE-DC+DC+8D8E4","O4B-2RAG+BAGFG","GFGEABO5C+O4A"
1300 PLAY "RDC+DO3A8O4F8.EDEO3A8O4G8","F4RAGAB-8B8O5C8C+8","RO3AO4D4L16GFGEB-A
G"
1310 PLAY "RF+EF+D8O5C8O4B-8A8B-AGA","D2.R8D8","ADO5C8.O4B-AB-AGF+GD8B-8"
1320 PLAY "RAGAD8G8.FEFO3B8O4E8","L8C+CO4BB-A4G+G","GF+EF+GFE-DC+8D8.CO3BO4C"
1330 PLAY "RDC+D8C+O3BO4C+","F+FE4D1.","D8O3G+8A4D1."
```

●演奏に表情をつける

(1) 

譜面(1)は，表情を特につけずに演奏する例です．これをPLAY命令のMMLで示すと次のようになります．

PLAY　"@2R8L16D8F+8D8A8D8>D8C+<BABAGF+GF+EL2DR4"

(2) 


譜面(1)に表情をつけてみたのが譜面(2)です．これをサウンド制御命令を使って表現してみると，次のようになります．

```
VOICE LFO 1,4,1,3500,−127,,5
                     ←─ⓒのための前設定

PLAY "@2R8＊L16Q1 @V115D8F＋8 @V120D8A8D8 !@V127Q2＞D8＊
           Ⓑ        Ⓐ                    Ⓒ  Ⓓ

Q6 @V120C＋＜BABA @V115GF＋G
    Ⓔ

F＋E Z10,2 Z15,8 Z20,4900 Z25,9 T80Q7 L2D."
    Ⓕビブラートをわかりやすくするために最後はテンポをおそくしています.
```

Ⓐクレッシェンド

徐々に音量が増す効果を付加します．実際には聴感上不自然でない程度に分割して音量を増加させます．

Ⓑスタカート

Q コマンドにより，音を短く区切ります．

Ⓒピッチベンド

LFO 変調効果によりワンショット波形でピッチ変調効果を付加します．効果が不必要な部分では＊コマンドで効果を停止させておき，！コマンドで，必要なところには効果を付加します．

Ⓓアクセント

アクセントは音量を一時的に増すことで表現します．

Ⓔデクレッシェンド

徐々に音量が減る効果を付加します．実際には聴感上不自然でない程度に分割して音量を減少させます．

Ⓕビブラート，トレモロなど

LFO 変調効果の繰り返し波形(三角波形)で，変調効果を付加します．

なお，この例の中の Z コマンドは次のような VOICE　LFO 命令と同等です．

Z10, 2Z15, 8Z20, 4900Z25, 9＝VOICE　LFO　X, 2, 8, 4900, 9

●基本音色を加工する

音色番号１の〝BRASS2〞の音色を利用して，犬の声を作ってみます．犬の声らしくするために，三角波ワンショット波形でピッチ変調をかけます．さらに，音程を低めに設定します．このように内蔵音色を加工して新しい音を創ることができます．

```
PLAY  "@1" : VOICE LFO 1, 5, 1, 4500, 127,,15
PLAY  "O3L8Q4CR {CCC} 4"
```

## ■外部オーディオ機器の使用

サウンド用外部オーディオ機器接続端子を使用することにより，外部オーディオ機器を接続することができます．

オーディオ機器への接続にはミニプラグを使用してください．ミニプラグを接続しますと，サウンド音は内部スピーカから出力されなくなりますが，BEEP音は内部スピーカから出力されます．

# 第12章

# ユーティリティ

## 12.1 BASICの登録

MS-DOS起動時に，BASIC(インタプリタ，コンパイラ)が自動的に起動するように設定することができます．これはユーザーが〝AUTOEXEX.BAT〟ファイルを編集することにより個別に行ってもかまいませんが，$N_{88}$-日本語BASIC(86)(MS-DOS版)では，質疑形式でこのようなディスクの作成を行うことができます．

コンパイラも同じ形式でディスクを作成することができます．

その方法を，フロッピィディスクに登録する場合と，固定ディスクに登録する場合に分けて解説します．

解説中で，BASICのディスクとは$N_{88}$-日本語BASIC(86)(MS-DOS版)，$N_{88}$-日本語BASIC(86)コンパイラの提供媒体またはそのバックアップのことを示します．

### ●フロッピィディスクに登録する場合

未使用のディスクを2枚用意します．このうち1枚は日本語入力用の辞書ファイルに使用しますので，日本語入力を使用しない場合やすでに辞書ファイルがある場合は1枚でかまいません．

①ドライブAからMS-DOSを起動します．

② MS-DOSの種類によって，次の2つの場合に操作方法が分かれます．

**・MS-DOSのメニュー画面に〝アプリケーションの登録〟が表示される場合**

[↑]キーや[↓]キーを使用して〝アプリケーションの登録〟にカーソルを移動し，[⏎]キーを押します．

次にシステムディスク(MS-DOSのシステムディスク：ドライブA)とアプリケーションディスク(BASICのディスク：ドライブB)のセットされているドライブ名を入力します．このとき，〝A：〟ではなく〝A〟というようにドライブ名の指定には〝：〟をつけないで指定します．

・**MS-DOSのメニュー画面に〝アプリケーションの登録〟が表示されない，またはメニュー画面がない場合**

ドライブBにBASICのディスクをセットして,MS-DOSのコマンドラインから次のように入力し，インストールプログラムを起動します.

A>B：INSTALL　B：　A：[↵]

(A>はMS-DOSのプロンプト)

③次の確認のメッセージが表示されますので,次に進む場合は[Y]キーを,登録を中止する場合は[N]キーを押します.

フロッピィディスクにN88-日本語BASIC(86)のシステムディスクを作成します.
よろしいですか(y/n)?

④初期化されるディスクをドライブBにセットします.

BASICシステムディスク用のフロッピィディスクを初期化しますか(y/n)?

ここで，[Y]キーを押すとBASICのシステムディスク用のディスクが初期化されます.
ディスクがすでにMS-DOSのシステムディスクとして作成されている場合などには,[N]キーを押します．この場合は⑥に進みます.
初期化が終了すると次のメッセージが表示されますので，[N]キーを押します.

別のディスクをフォーマットしますか (y/n)?

⑤使用する日本語入力方式を選択します.次の画面が表示されますので,1から6の番号のキーをタイプします.

1. AI逐次変換入力
2. AI連文節変換入力
3. 逐次変換入力
4. 連文節変換入力
5. 文節変換入力
6. 使用しない

選択できる番号はMS-DOSの種類によって異なりますので,詳しくは日本語入力ガイドを参照してください.
番号を入力するとMS-DOSのシステムディスクから日本語入力フロントエンドプロセッサがコピーされます.

市販の日本語入力フロントエンドプロセッサを使用する場合には，6をタイプして，後でフロントエンドプロセッサを組み込んでください．

⑥次のメッセージが表示されます．

ドライブA：にN88-日本語BASIC(86)オリジナルディスクを
ドライブB：にBASICを登録するディスクをセットして下さい．

ドライブAにBASICのディスクを，ドライブBにBASIC登録用のディスクをセットし，[⏎]キーを押します．
ドライブA内の必要なファイルがドライブBにコピーされます．コピーが終了すると次のメッセージが表示されます．

ドライブA：にMS-DOSのシステムディスクをセットして下さい．

MS-DOSのシステムディスクをドライブAにセットして，[⏎]キーを押してください．必要なファイルのコピーが終わると，BASICのシステムディスクの作成は完了です．

⑦以降は，⑤で6以外のキーを押した場合の処理です．

⑦次のメッセージが表示されます．

辞書ディスクを作成しますか（y/n）？

BASIC専用の辞書ファイルを使用したい場合には，ディスクの初期化が行われますので，辞書用のディスクをドライブBにセットして[Y]キーを押します．
すでに用意されている辞書ファイルを使用する場合には[N]キーを押します．
初期化が終わると次のメッセージが表示されますので，[N]キーを押して，[⏎]キーを押します．

別のディスクをフォーマットしますか（y/n）？

⑧次のメッセージが表示されますので，ドライブAにMS-DOSの辞書ファイルが入っているディスクをセットします．

ドライブA：にMS-DOSの辞書ディスク(マスタ)をセットして下さい．

[⏎]キーを押すと，ドライブAの辞書ファイルがドライブBにコピーされます．
このとき，日本語入力フロントエンドプロセッサと辞書ファイルとが対応していない場合，エラーメッセージが表示されて，再度確認入力待ちとなります．この場合，対応する辞書ファイルをドライブAにセットしてください．
コピーが終了すると辞書ファイルの完成です．

これ以降，BASICのシステムディスクをドライブAに，辞書ディスクをドライブBにセットしてシステムを起動すると，MS-DOS起動時に自動的にBASIC(インタプリタ，コンパイラ)が起動されます．

ただし，④でディスクを初期化しなかった場合には，BASICは自動的に起動されません．初期化すると登録したインタプリタまたはコンパイラが自動的に起動されます．

**●固定ディスクに登録する場合**

固定ディスク内のMS-DOSのメニューにBASIC(インタプリタ，コンパイラ)を登録します．

固定ディスクはあらかじめMS-DOSで初期化されていて，MS-DOSのシステムが組み込まれている必要があります．固定ディスクの初期化などについてはMS-DOSのマニュアルを参照してください．

①固定ディスクからMS-DOSを起動します．

② MS-DOSの種類によって，次の2つの場合に操作方法が分かれます．

**・MS-DOSのメニュー画面に〝アプリケーションの登録〟が表示される場合**

[↑]キーや[↓]キーを使用して〝アプリケーションの登録〟にカーソルを移動し，[⏎]キーを押します．

次にシステムディスク(MS-DOSを起動した固定ディスク)とアプリケーションディスク(BASICのディスク：ドライブB)のセットされているドライブ名を入力します．このとき，〝A：〟ではなく〝A〟というようにドライブ名の指定には〝：〟をつけないで指定します．

**・MS-DOSのメニュー画面に〝アプリケーションの登録〟が表示されない，またはメニュー画面がない場合**

ドライブBにBASICのディスクをセットして，MS-DOSのコマンドラインから次のように入力し，インストールプログラムを起動します．

```
A>B：INSTALL　B：　A：[⏎]
```

(A>はMS-DOSのプロンプト)

③次の確認のメッセージが表示されますので，次に進む場合は[Y]キーを，登録を中止する場合は[N]キーを押します．

```
固定ディスクにN88-日本語BASIC(86)を登録します．
よろしいですか(y/n)？
```

④ドライブB内の必要なファイルがドライブA内の¥BASICディレクトリ内にコピーされます．作業終了後，次のメッセージが表示されます．

N88-日本語BASIC(86)を登録しました

これ以降，固定ディスクからMS-DOSを起動するとメニューの選択によりBASICが起動します．

メニューを使用しない場合は，ディレクトリを¥BASICに移動して通常の起動方法で，BASICを起動します．

## 12.2 セットアップユーティリティ

$N_{88}$-日本語BASIC(86)(MS-DOS版)および$N_{88}$-日本語BASIC(86)ランタイムサポートパッケージ(ランタイムライブラリ)の提供媒体には，セットアップユーティリティ(N88SETUP.EXE)が格納されています．これは画面ハードコピー機能，演算モードの設定や，RS-232Cの初期設定を行うためのものです．

▶参照　画面ハードコピー→「5.3　画面ハードコピー」

セットアップユーティリティはインタプリタ本体(N88BASIC.EXE)またはランタイムライブラリ(N88BASIC.LIB)の内容を更新します．セットアップユーティリティで一度設定した値は，再び値を変更するまで有効です．設定の対象になるファイルは書き込み可能状態(ライトプロテクトシールが貼られていないなど)にしておいてください．

---

**注意**：セットアップユーティリティは，ファイルの内容を更新してしまいますので，提供媒体の中のファイルを他のディスクにコピーし，コピーしたものに対してセットアップを行うようにしてください．

---

セットアップユーティリティの使用方法は次のとおりです．

①MS-DOSが起動されている状態で，セットアップユーティリティの格納されているディスクをカレントドライブにセットし，次のように入力します．

A>N88SETUP [↵]

(A>はMS-DOSのプロンプト)

②セットアップユーティリティが起動され，次のような画面が表示されます．

```
N88－日本語BASIC（86）セットアップユーティリティ
セットアップ対象ファイル名は（N88BASIC.EXE N88BASIC.LIB）？
█

RS－232Cの初期設定のみ行なう場合には省略できます
```

ここでセットアップの対象となるファイル名，すなわちインタプリタシステムの場合には“N88BASIC.EXE”，コンパイラシステムの場合には“N88BASIC.LIB”を指定して，[⏎]キーを押します．

なお，ここでドライブ名やディレクトリ名を付加した指定も可能です．

例）　C：¥BIN¥BASIC¥N88BASIC.EXE

また，RS-232Cの初期設定のみを指定する場合はファイル名を入力する必要はありません．[⏎]キーのみを押してください．

③ファイル名を入力すると次の基本メニューが表示されます．

```
N88－日本語BASIC（86）セットアップユーティリティ
［基本メニュー］

画面ハードコピーの機能指定
演算モードの設定
RS－232Cの初期設定
終　　了

↓キー，↑キーで選択してRETURNキーを押すとその処理に移ります
注意：終了までディスクを入れ換えないでください
```

反転しているところは現在選択されている機能を表しています．

カーソル移動キー(↑，↓)を使用して反転表示しているカーソルを移動し，⏎キーを押すと，カーソルが示しているサブメニューの処理に移行します．ただし，対象ディスクの種類と選択した項目の組み合せによっては設定できない場合があり，このときはスピーカが鳴り，基本メニューが再表示されます．

STOPキーを押すと基本メニューが再表示されます．

**画面ハードコピーの機能設定**……⏎キーを押すと④へ進む
**演算モードの設定**……………………⏎キーを押すと⑦へ進む
**RS-232Cの初期設定**……………⏎キーを押すと⑧へ進む
**終了**…………………………………⏎キーを押すと⑨へ進む

④画面ハードコピーの機能設定

次のモード選択メニューが表示されます．

N88－日本語BASIC (86) セットアップユーティリティ
［画面ハードコピーの機能設定］　　（モード選択）
ノーマルモード用
ハイレゾリューションモード用
基本メニューに戻る
↓キー，↑キーで選択してRETURNキーを押してください

反転しているところは現在選択されている機能を表しています．メニュー内での操作は，③の基本メニューの場合とまったく同様です．

**ノーマルモード用**…………………⏎キーを押すと⑤へ進む
**ハイレゾリューションモード用**…⏎キーを押すと⑥へ進む
**基本メニューにもどる**……………⏎キーを押すと③へもどる

⑤ノーマルモード用画面ハードコピー機能の設定

次のメニューが表示されます.

```
N88－日本語BASIC（86）セットアップユーティリティ
[画面ハードコピーの機能設定]　（ノーマルモード用）

機　能　　　　　　　　　　　：　モノクロコピー　カラーコピー　N88-BASIC(86)互換

COPY　1　(CTRL+COPY キー)　方向　：　水平　垂直
COPY　2　(GRPH+COPY キー)　倍率　：　標準　1/3倍　2/3倍　3/3倍　4/3倍
　　　　　　　　　　　　　方向　：　水平　垂直
COPY　3　(COPY キー)　　　倍率　：　標準　1/3倍　2/3倍　3/3倍　4/3倍
　　　　　　　　　　　　　方向　：　水平　垂直
↓キー，↑キー，←キー，→キーで選択してRETURNキーを押してください
設定を中断する場合はSTOPキーまたはCTRL－Cキーを押してください
```

反転しているところは現在選択されている機能を表しています.

カーソル移動キー(↑, ↓, ←, →)を使用して反転表示しているカーソルを移動し,

⏎キーを押すと, 選択を終了して確認入力の画面に進みます.

STOPキーを押すと④のモード選択メニューにもどります. このとき, 設定の変更は無効になります.

選択する項目と実際の画面ハードコピーとの対応は次のとおりです. それ以外の項目の組み合せは選択できませんので注意してください.

| 画面ハードコピー | 機　能 | COPY 1～3 | |
|---|---|---|---|
| | | 方向 | 倍率 |
| 拡大/縮小コピーおよび印字方向設定機能 | モノクロコピー | 水平, 垂直 | 標準, 1/3～4/3 |
| 通常の画面ハードコピー機能 | | 水平のみ | 標準のみ |
| カラー画面ハードコピー機能 | カラーコピー | | |
| $N_{88}$-日本語 BASIC(86)の通常のハードコピー機能と同等の機能 | $N_{88}$-BASIC(86)互換 | | |

・確認入力

```
N88－日本語BASIC（86）セットアップユーティリティ
[画面ハードコピーの機能設定]　（ノーマルモード用）

機　能                                 ：モノクロコピー　カラーコピー　N88-BASIC(86)互換

COPY 1 (CTRL+COPY キー) 方向 ：水平 垂直
COPY 2 (GRPH+COPY キー) 倍率 ：標準 1/3倍 2/3倍 3/3倍 4/3倍
                        方向 ：水平 垂直
COPY 3 (COPY キー)      倍率 ：標準 1/3倍 2/3倍 3/3倍 4/3倍
                        方向 ：水平 垂直

変更してもよろしいですか（y/n）？
```

選択に誤りがなければ Y キーを入力します．すると②で指定したファイルに新しい情報が書き込まれ，④のモード選択メニューにもどります．

選択に誤りがあれば N キーを入力します．すると⑤の最初の画面にもどって再び機能の選択をすることができます．

---

**注意**：印字方向の「水平」，印字倍率の「標準」以外の値を設定した場合，PC-PR601系ページプリンタ以外のプリンタ(たとえばPC-PR201系)を接続すると画面ハードコピーが正常に動作しませんので注意してください．

カラーハードコピーの「カラーコピー」を設定した場合，PC-PR201V系プリンタを接続してください．

---

⑥ハイレゾモード用画面ハードコピー機能の設定

次のメニューが表示されます．

```
N88－日本語BASIC（86）セットアップユーティリティ
[画面ハードコピーの機能設定]　（ハイレゾリューションモード用）
COPY 1 (CTRL+COPY キー) 方向 ：水平 垂直
COPY 2 (GRPH+COPY キー) 倍率 ：標準 2/3倍 4/3倍
                        方向 ：水平 垂直
COPY 3 (COPY キー)      倍率 ：標準 2/3倍 4/3倍
                        方向 ：水平 垂直

↓キー，↑キー，←キー，→キーで選択してRETURNキーを押してください
設定を中断する場合はSTOPキーまたはCTRL－Cキーを押してください
```

反転しているところは現在選択されている機能を表しています.

カーソル移動キー( ↑ , ↓ , ← , → )を使用して反転表示しているカーソルを移動し, ⏎ キーを押すと, 選択を終了して確認入力の画面に進みます.

STOP キーを押すと④のモード選択メニューにもどります. このとき, 設定の変更は無効になります.

・**確認入力**

```
N88－日本語ＢＡＳＩＣ（86）セットアップユーティリティ
［画面ハードコピーの機能設定］　　（ハイレゾリューションモード用）

COPY 1 （CTRL+COPY キー） 方向 ： 水平 垂直
COPY 2 （GRPH+COPY キー） 倍率 ： 標準 2/3倍 4/3倍
                          方向 ： 水平 垂直
COPY 3 （COPY キー）      倍率 ： 標準 2/3倍 4/3倍
                          方向 ： 水平 垂直

変更してもよろしいですか（y/n）？█
```

選択に誤りがなければ Y キーを入力します. すると②で指定したファイルに新しい情報が書き込まれ, ④のモード選択メニューにもどります.

選択に誤りがあれば N キーを入力します. すると⑥の最初の画面にもどって再び機能の選択をすることができます.

---

**注意**：印字方向の「水平」, 印字倍率の「標準」以外の値を設定した場合, PC-PR601系ページプリンタ以外のプリンタ(たとえばPC-PR201系)を接続すると画面ハードコピーが正常に動作しませんので注意してください.

---

⑦演算モードの設定

次のメニューが表示されます．

```
N88－日本語BASIC（86）セットアップユーティリティ
 ［演算モードの設定］
演算モード　：　基本演算モード　拡張演算モード

←キー，→キーで選択してRETURNキーを押してください
設定を中断する場合はSTOPキーまたはCTRL－Cキーを押してください
```

BASICでは通常の演算モード(基本演算モード)のほかに,拡張演算モードを使用することができます．拡張演算モードとは基本演算モードとは異なる演算方式を使用することにより，単精度実数の平方根(SQR)演算およびべき乗演算(べき数が整数の場合のみ)が高速化されたモードです．

---

**注意**・拡張演算モードと基本演算モードでは，その演算結果がわずかに異なる場合があります．
・ハイレゾモードでは，演算は常に拡張演算モードで行われます．Ⓗ

---

反転しているところは現在選択されている機能を表しています．

カーソル移動キー(［←］，［→］)を使用して反転表示しているカーソルを移動し，［⏎］キーを押すと，選択を終了して確認入力の画面に進みます．

［STOP］キーを押すと③の基本メニューにもどります．このとき，設定の変更は無効になります．

・**確認入力**

```
N88－日本語BASIC（86）セットアップユーティリティ
 ［演算モードの設定］
演算モード　：　基本演算モード　拡張演算モード

変更してもよろしいですか（y/n）？■
```

選択に誤りがなければ Y キーを入力します．すると②で指定したファイルに新しい情報が書き込まれ，③の基本メニューにもどります．

選択に誤りがあれば N キーを入力します．すると⑦の最初の画面にもどって再び機能の選択をすることができます．

⑧ RS-232C の初期設定

次のメニューが表示されます．

```
        N88－日本語BASIC（86）セットアップユーティリティ
     [RS－232Cの初期設定]  （メモリスイッチの設定）
ボーレート              ：  75  150  300  600  1200  2400  4800  9600
データbit長             ：  7bit  8bit
ストップbit長           ：  1bit  1.5bit  2bit
パリティチェック        ：  使わない  奇数  偶数
Xパラメータ             ：  無効  有効
Sパラメータ             ：  無効  有効
DELコード受信時動作     ：  BS/DEL  NUL/NUL
↓キー，↑キー，←キー，→キーで選択してRETURNキーを押してください
設定を中断する場合はSTOPキーまたはCTRL－Cキーを押してください
```

反転しているところは現在選択されている機能を表しています．

カーソル移動キー（ ↑ ， ↓ ， ← ， → ）を使用して反転表示しているカーソルを移動し， ⏎ キーを押すと，選択を終了して確認入力の画面に進みます．

STOP キーを押すと③の基本メニューにもどります．このとき，設定の変更は無効になります．

・確認入力

```
        N88－日本語BASIC（86）セットアップユーティリティ
      ［RS－232Cの初期設定］  （メモリスイッチの設定）
ボーレート           ： 75 150 300 600 1200 2400 4800 9600
データbit長          ： 7bit 8bit
ストップbit長        ： 1bit 1.5bit 2bit
パリティチェック     ： 使わない 奇数 偶数
Xパラメータ          ： 無効 有効
Sパラメータ          ： 無効 有効
DELコード受信時動作  ： BS/DEL NUL/NUL
変更してもよろしいですか（y/n）？
```

選択に誤りがなければ Y キーを入力します．すると本体のメモリスイッチに新しい情報が書き込まれ，③の基本メニューにもどります．

選択に誤りがあれば N キーを入力します．すると⑧の最初の画面にもどって再び機能の選択をすることができます．

⑨終了

属性を変更した場合は，〝属性を変更しました〟というメッセージを表示して MS-DOS に制御をもどします．

属性を変更しなかった場合は，〝属性は変更しませんでした〟というメッセージを表示して MS-DOS に制御をもどします．

●セットアップユーティリティのエラーメッセージ

セットアップユーティリティ実行中に異常が発生した場合，次のエラーメッセージが表示されます．

・ファイル名の指定が不正です．

指定したファイル名が〝N88BASIC．EXE〟または〝N88BASIC．LIB〟以外か，ドライブ名またはパス名の指定が正しくありません．

正しいファイル名を指定してください．

**・ファイルが見つかりません.**

指定したファイルが見つかりません. ドライブの指定が誤っているかまたはそのドライブに目的のファイルが格納されていないなどの理由が考えられます. ディスクの内容を確認してください.

**・このファイルはセットアップできません.**

指定した "N88BASIC.EXE" または "N88BASIC.LIB" とセットアップユーティリティの対応が取れていません. "N88BASIC.EXE" または "N88BASIC.LIB" の提供媒体内にあるセットアップユーティリティ(N88SETUP.EXE)を使用してください.

**・ファイルの更新に失敗しました**

ファイルが書き込み禁止状態になっていたり, ファイルがこわれていたりしたことによりファイルを更新することができません. ライトプロテクトシールやファイルを再確認した後, 再度設定してください.

## 12.3 ファイル変換ユーティリティ

$N_{88}$-日本語 BASIC(86)で作成された各種ファイルは, ファイル変換ユーティリティで MS-DOS 形式のファイルに変換して, MS-DOS 上のディスクにコピーすることにより $N_{88}$-日本語 BASIC(86)(MS-DOS 版)で利用することができます.

日本語を含まないファイルについては, $N_{88}$-日本語 BASIC(86)で作成したすべてのファイルを MS-DOS 版に変換できます.

次に示すファイルについてはファイル変換ユーティリティで日本語コード変換指定を行った場合に限り, 日本語コードを含むファイルでも変換が可能です.

・アスキー形式プログラムファイル
・シーケンシャルデータファイル
・ランダムデータファイル

---

**参考**: 日本語の内部コード(日本語表現)は, $N_{88}$-日本語 BASIC(86)(MS-DOS 版)はシフト JIS で, $N_{88}$-日本語 BASIC(86)は JIS コードです. このため, 日本語を含むファイルは単にファイル形式のみを変換しても, ファイルの内容が正確に変換できません.

---

日本語コードを含むバイナリ形式のプログラムファイルは，一度アスキー形式でセーブし直してから変換しなければなりません．

日本語コードを含むプロテクトセーブ形式のプログラムファイルは変換できません．

ファイル変換ユーティリティの起動はMS-DOSが起動されている状態で，"FILECONV.EXE"の格納されているMS-DOSのシステムディスクをカレントドライブにセットし，次のように入力します．

A>FILECONV [↵]

(A>はMS-DOSのプロンプト)

ファイル変換ユーティリティが起動されますので，メッセージに従い操作します．

▶**参照**　ファイル変換ユーティリティ→「MS-DOSのマニュアル」

プログラムファイルを変換する際の注意点を次に示します．

・次の命令はBASICでサポートされていません．削除した後にファイル変換を行ってください．

| | |
|---|---|
| ON PEN GOSUB | MON |
| PEN ON/OFF/STOP | TERM |
| PEN | LOAD? |
| DSKI$ | OPEN "CAS:" |
| DSKO$ | MOTOR |
| FPOS | NEW ON |

・$N_{88}$-日本語BASIC(86)とBASICでは次のような中間コード(バイナリ形式のプログラムファイルのコード)の意味が異なります．

| 中間コード | $N_{88}$-日本語BASIC(86) | $N_{88}$-日本語BASIC(86)(MS-DOS版) |
|---|---|---|
| &HFFC1 | DSKI$ | ENVIRON$ |
| &H97 | DSKO$ | ENVIRON |
| &HB6 | MON | CHILD |
| &HB4 | MOTOR | SCROLL |
| &HFF83 | PEN | MOUSE |
| &HDB | TERM | SYSTEM |
| &HFFA3 | FPOS | SEGPTR |

ファイル変換ユーティリティはバイナリ形式プログラムファイルの変換の際，これらのコードに対して特に処理を行いません．

したがって，これらの命令を削除しないで変換した場合，命令が変わってしまいますので注意してください．

・$N_{88}$-日本語 BASIC(86)で作成された機械語プログラムファイルの先頭には制御用データとして4バイトの情報が付加されています．

　$N_{88}$-日本語 BASIC(86)(MS-DOS 版)では，この4バイトの情報は必要としませんので，ファイル変換ユーティリティは無条件に機械語プログラムファイルの先頭4バイトを削除します．

・変換したファイルをハイレゾモードで使用する場合，テキスト画面に40桁モードと20行モードがないため，テキスト画面に対する WIDTH 命令は無効な命令となります．

・変換したファイルをハイレゾモードで使用する場合には，グラフィック画面の解像度(グラフィック画面のサイズ)が異なりますので，グラフィック関連命令の座標系を見直す必要があります．

# 第13章

# チャイルドプロセス制御機能

MS-DOSでは，MS-DOS上で動作中のあるプログラム(ここではBASIC)から，そのプログラムの実行を終了することなくMS-DOS上の他のプログラムを起動し実行することのできる，〝チャイルドプロセス制御機能〟がサポートされています．

この場合，BASICを親プロセス，他のプログラムを子プロセスまたはチャイルドプロセスと呼びます．

チャイルドプロセス制御機能により，BASICの環境から，MS-DOSが提供する種々のコマンド(内部コマンド，外部コマンド，各種ユーティリティ)の機能を使用することが可能になります．

BASICで提供されているチャイルドプロセス制御に関連する命令は，次のとおりです．

CHILD命令 …………チャイルドプロセスを起動する
ENVIRON命令 ………環境文字列テーブルを管理する
ENVIRON$関数 ……環境文字列テーブルから情報を抽出する

次に各命令の内容を解説します．

▶参照　MS-DOSのコマンド，用語→「MS-DOSのマニュアル」

## 13.1 チャイルドプロセスの起動

CHILD 命令により BASIC の環境から一時的に MS-DOS の環境に制御を移すことができます.

### CHILD 命令

CHILD　〔<チャイルドプロセス名およびチャイルドプロセスに引き渡す情報>〕
〔, M〕〔, <画面クリアスイッチ 1>〕〔, <画面クリアスイッチ 2>〕
〔, <ファンクションキー表示スイッチ>〕

**・<チャイルドプロセス名およびチャイルドプロセスに引き渡す情報>**

起動するチャイルドプロセスの名前(プログラム名)とチャイルドプロセスに引き渡す情報を124 文字以内の文字列で指定します. 基本的な指定形式は次のとおりです.

"<チャイルドプロセス名> △〔<引き渡し情報>〕"

(△は空白を意味します)

<チャイルドプロセス名> は起動するプログラム名で, 〔<ドライブ名>〕<プログラム名> の形で指定します.

<引き渡し情報> は, 起動するプログラムに必要な任意の文字列(パラメータなど)です.

プログラムの実行が終了すると自動的に制御は MS-DOS から BASIC にもどります.

<チャイルドプロセス名およびチャイルドプロセスに引き渡す情報>を省略すると, MS-DOS に制御が移行された後, プロンプトメッセージが出力され, コマンド待ちの状態になります. この場合, BASIC に制御をもどすためには EXIT コマンド(MS-DOS の内部コマンド)を実行します.

**・M**

M を指定しない場合, チャイルドプロセスはチャイルドプロセス領域にロードされます. したがって, チャイルドプロセス領域を CLEAR 命令によりあらかじめ確保しておく必要があります. 指定するチャイルドプロセス領域のサイズは次のとおりです.

<MS-DOS の COMMAND.COM のサイズ>+<起動するプログラムのサイズ>

チャイルドプロセス終了後, BASIC プログラムの処理を続行することができます. ただし, すべてのファイルがクローズされるので, 必要な場合はファイルを再びオープンする必要があります.

Mを指定した場合，チャイルドプロセスはシンボルテーブル領域以降にロードされ，実行されます．したがってチャイルドプロセス領域を用意する必要はありません．

Mを指定するとBASICプログラムの処理は続行できなくなり，すべての変数／配列の値は初期化されます．ただし，BASICプログラムそのものはこわされることはありません．

コンパイラの場合，チャイルドプロセス終了後にMS-DOSにもどります．

・<画面クリアスイッチ1>

BASICからチャイルドプロセス(MS-DOS)に制御を移行する際，BASICの画面の状態を次の値で指定します．

| | |
|---|---|
| 0または省略 | ：画面を消去する．このときのカーソルの位置はホームポジション(左上隅) |
| 1 | ：画面はそのまま残す．このときのカーソルの位置は最下行(左下隅) |

・<画面クリアスイッチ2>

チャイルドプロセス(MS-DOS)からBASICに制御をもどす際，チャイルドプロセスの画面の状態を次の値で指定します．

| | |
|---|---|
| 0または省略 | ：画面はそのまま残す．このときのカーソルの位置は最下行(左下隅) |
| 1 | ：画面を消去する．このときのカーソルの位置はホームポジション(左上隅) |

・<ファンクションキー表示スイッチ>

チャイルドプロセス起動時にMS-DOSのファンクションキーの表示状態を次の値で指定します．

| | |
|---|---|
| 0または省略 | ：ファンクションキーを表示する |
| 1 | ：ファンクションキーを表示しない |

例1）　CHILD　" COPY　SUB1.BAS　B:", M

カレントドライブのカレントディレクトリにあるファイル"SUB1.BAS"をドライブBのカレントディレクトリにコピー(COPYコマンドはMS-DOSの内部コマンド)します．コピー終了後，制御は自動的にBASICにもどります．

例2）　CHILD, M

MS-DOSのプロンプトメッセージが表示され，MS-DOSに対するコマンド待ちの状態になります．以降，ユーザーが直接MS-DOSを操作することが可能となります．BASICに制御をもどすためにはEXITコマンド(MS-DOSの内部コマンド)を実行します．

---

**注意**：チャイルドプロセス制御を行う場合，次の点に注意してください．

- CHILD 命令を実行する場合，カレントドライブにセットされているディスクのルートディレクトリ下にコマンドプロセッサ(COMMAND．COM)が用意されていない場合，〝CHILD　can't execute〟エラーとなります．
- チャイルドプロセスに制御が移行された後の MS-DOS の画面サイズは，BASIC の画面サイズにかかわらず無条件に 80 桁，25 行モードになります．

---

▶参照　CHILD 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

## 13.2　環境文字列テーブルの管理

ENVIRON 命令は BASIC の環境文字列テーブルを更新します．環境文字列とは各コマンド，アプリケーションが参照可能な文字列のことです．この文字列は BASIC 自身には何の意味もありませんが，BASIC から起動されたチャイルドプロセスに引き継がれます．

ENVIRON 命令

ENVIRON　〈文字列〉

環境文字列テーブルは〝名前＝パラメータ〟を基本単位とする文字列の集まりです．ENVIRON 命令に指定する〈文字列〉はこの形式で指定します．

環境文字列テーブル内の名前は一般に任意の名前で，〝PATH〟以外の名前はチャイルドプロセスが独自に意味付けを行います(MS-DOS のシステムでは意味を持ちません)．

ただし〝PATH＝〟に続く文字列は，チャイルドプロセスが起動される際のディレクトリ情報として参照されます．

例）ENVIRON　" PATH=¥BASIC"
　　CHILD　" A：SUB1"

ドライブ A のディレクトリ¥BASIC の SUB1 が起動されます．

▶参照　ENVIRON 命令→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

## 13.3 環境文字列テーブルの参照

ENVIRON$関数は環境文字列テーブル内の指定されたパラメータの値を得る関数です．

**ENVIRON $ 関数**

1） ENVIRON$(<名前>)

2） ENVIRON$(<番号>)

1)のように〈名前〉を指定すると，環境文字列テーブル内から名前の一致する環境文字列のパラメータがとり出されます．

2)のように〈番号〉を指定すると，環境文字列テーブル内の〈番号〉番目の環境文字列がそのままとり出されます．

例1） A$=ENVIRON$(" PATH")

PATHのパラメータを得ます．

例2） A$=ENVIRON$(2)

2番目のパラメータの値を得ます．

▶参照 ENVIRON$関数→「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」

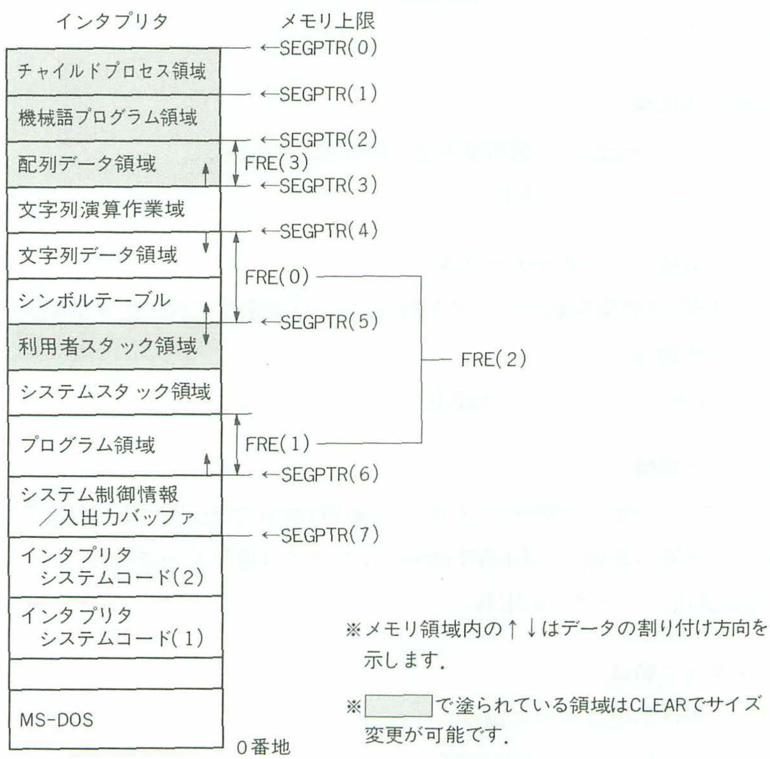
# 第14章

# メモリマップとメモリ管理

BASICのプログラムを実行する際のメモリマップとメモリ管理について解説します．

## 14.1　メモリマップとメモリレイアウトの変更

### ■インタプリタの場合



・**チャイルドプロセス領域**

チャイルドプロセス用の領域．CLEAR命令によりサイズ変更が可能です．

CHILD命令のパラメータにM指定がある場合，この領域は必要としません．

BASIC起動時のサイズ：0

▶**参照**　チャイルドプロセス→「第13章　チャイルドプロセス制御機能」

各命令の機能→「$N_{88}$-日本語BASIC(86)(MS-DOS版)リファレンスマニュアル」

・**機械語プログラム領域**

機械語プログラムロード用の領域．CLEAR命令によりサイズ変更が可能です．

BASIC起動直後のサイズ：0

・**配列データ領域**

配列制御情報および数値型配列データの格納領域．CLEAR命令によりサイズ変更が可能です．

BASIC起動直後のサイズ：文字列演算用作業域以降メモリの上限まで．CLEAR命令によりサイズ変更が可能です．

・**文字列演算用作業域**

文字列の演算(文字列操作)に使用される一時作業用の領域．

BASIC起動直後のサイズ：2KB

・**文字列データ領域／シンボルテーブル**

変数の制御情報(数値型変数のデータを含む)および文字型変数／文字型配列に代入される文字列データの格納領域．

BASIC起動直後のサイズ：最大62KB

・**利用者スタック領域**

利用者プログラム実行制御用のスタック領域(演算処理／FOR～NEXT命令／GOSUB～RETURN命令等の制御)．CLEAR命令によりサイズ変更が可能です．

BASIC起動直後のサイズ：0.5KB

・**システムスタック領域**

インタプリタの作業用スタック領域．

BASIC起動直後のサイズ：1KB固定

・**プログラム領域**

プログラム格納領域．

BASIC 起動直後のサイズ：システム制御情報／入出力バッファ，利用者スタック，およびシステムスタックを含め 64KB

・**システム制御情報**

インタプリタ用各種制御情報.

・**入出力バッファ**

ディスクファイルアクセス用バッファ.

BASIC 起動直後のサイズ：インタプリタ起動時のファイル同時オープン数×バッファサイズ分(起動時に自動的に確保される)

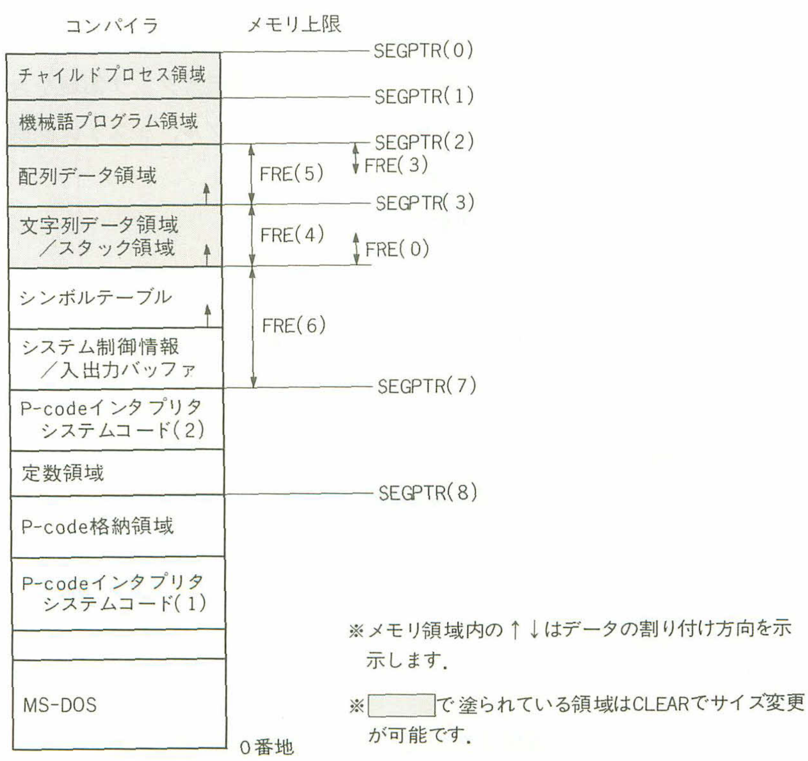
・**システムコード(2)**

インタプリタ拡張部.

・**システムコード(1)**

インタプリタ本体.

## ■コンパイラの場合



**・チャイルドプロセス領域**

チャイルドプロセス用の領域．CLEAR 命令によりサイズ変更が可能です．

CHILD 命令のパラメータに M 指定がある場合，この領域は必要としません．

プログラム起動時のサイズ：0

▶**参照**　チャイルドプロセス→「第 13 章　チャイルドプロセス制御機能」

各命令の機能→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

**・機械語プログラム領域**

機械語プログラムロード用の領域．CLEAR 命令によりサイズ変更が可能です．

プログラム起動直後のサイズ：0

**・配列データ領域**

配列制御情報および数値型配列データの格納領域．CLEAR 命令によりサイズ変更が可能です．

プログラム起動直後のサイズ：文字列データ領域／スタック領域以降メモリの上限まで．

**・文字列データ領域／スタック領域**

インタプリタの文字列データ領域と各種スタック領域を合わせたものです．

文字列データ領域は文字型変数／文字型配列に代入される文字列データの格納領域です．

スタック領域は，利用者プログラム実行制御用のスタック領域(演算処理／FOR～NEXT 命令／GOSUB～RETURN 命令等の制御)とインタプリタの作業用スタック領域を合わせたものです．

なお，文字列データ領域／スタック領域，シンボルテーブル，システム制御情報／入出力バッファは合わせて 64KB まで確保できます．

CLEAR 命令によりサイズ変更が可能です．

**・シンボルテーブル**

変数の制御情報(数値型変数のデータを含む)の格納領域．

プログラム起動直後のサイズ：コンパイル時に決定され，以降不変です．

**・システム制御情報／入出力バッファ**

システム制御情報はプログラム用各種制御情報です．

入出力バッファはディスクファイルアクセス用バッファです．

入出力バッファのサイズはファイル同時オープン数×バッファサイズ分(REM　$FILE 命令で指定)です．

**・定数領域**

プログラム中で使用される数値定数／文字列定数の格納領域です．

プログラム起動直後のサイズはプログラムの構造に依存します．

・P-code インタプリタシステムコード(2)

P-code インタプリタ拡張部.

・P-code 格納領域

プログラム(コンパイラにより P-code 化されている)格納領域.

プログラム起動直後のサイズはプログラムの命令形式／命令数に依存しますが，最大で 4×64KB です.

・P-code インタプリタシステムコード(1)

P-code インタプリタ本体.

## 14.2 セグメントベースの宣言

セグメントのベースアドレスを宣言します．書式は次のとおりです.

DEF SEG 命令

DEF SEG= 〈セグメントアドレス〉

〈セグメントアドレス〉はセグメントのベースアドレスを指定するので，実際のアドレスを 16 で割った値です.

BASIC との各制御領域の先頭ベースアドレスは SEGPTR 関数で得ることができます.

## 14.3 メモリレイアウトの参照

SEGPTR 関数は各領域のセグメントベースを得ます.得られる値はセグメントベースですので，実際のアドレスを 16 で割った値です．SEGPTR 関数の内容を次の表に示します.

| 機能 | インタプリタ | コンパイラ |
|---|---|---|
| SEGPTR(0) | メモリの上限 | メモリの上限 |
| SEGPTR(1) | チャイルドプロセス領域の先頭 | チャイルドプロセス領域の先頭 |
| SEGPTR(2) | 機械語プログラム領域の先頭 | 機械語プログラム領域の先頭 |
| SEGPTR(3) | 配列データ領域の先頭 | 配列データ領域の先頭 |
| SEGPTR(4) | 文字列データ領域/シンボルテーブルの終端 | 使用できない |
| SEGPTR(5) | 文字列データ領域/シンボルテーブルの先頭 | |
| SEGPTR(6) | プログラム領域の先頭 | |
| SEGPTR(7) | システム制御情報領域の先頭 | システム制御情報領域の先頭 |
| SEGPTR(8) | 使用できない | 定数領域の先頭 |

## 14.4　領域のサイズ

FRE関数はメモリの使用状況をメモリサイズによって得ます．FRE関数の内容を次の表に示します．

| 機能 | インタプリタ | コンパイラ |
|---|---|---|
| FRE(0) | 文字列データ領域/シンボルテーブルの未使用サイズ | 文字列データ領域/スタック領域の未使用サイズ |
| FRE(1) | プログラム領域の未使用部分のサイズ | 使用できない |
| FRE(2) | FRE(0)とFRE(1)の合計サイズ | |
| FRE(3) | 配列データ領域の未使用部分のサイズ | 配列データ領域の未使用部分のサイズ |
| FRE(4) | 使用できない | 文字列データ領域/スタック領域の全体サイズ |
| FRE(5) | | 配列データ領域の全体サイズ |
| FRE(6) | | シンボルテーブルとシステム制御情報/入出力バッファ領域の合計サイズ |

## 14.5　変数格納アドレス

VARPTR関数の書式は次のとおりです．

**VARPTR関数**

```
VARPTR( |    〈変数名〉     |〔,〈機能〉〕)
        | #〈ファイル番号〉 |
```

〈変数名〉を指定した場合は，その変数または配列変数のデータが格納されている領域のアドレスが得られます．

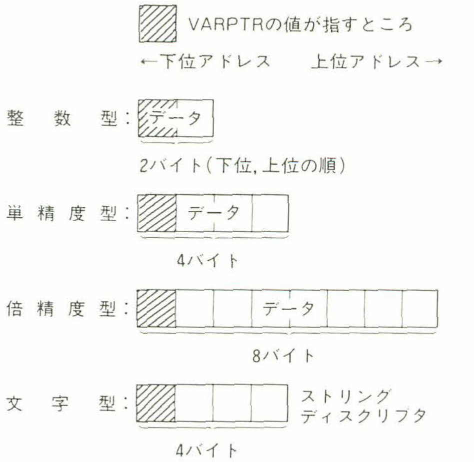
〈ファイル番号〉を指定した場合は，そのファイル番号に割り当てられているファイルのFCB(ファイルコントロールブロック)のアドレスが得られます．

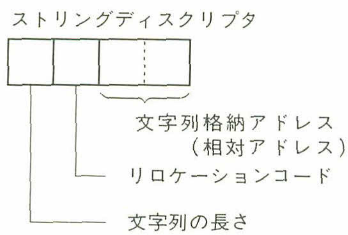
〈機能〉の内容は次のとおりです．

0：セグメントベースからの相対アドレスを得る．

1：セグメントベースを得る．

セグメントベース相対アドレスが指示するアドレスはデータの型によりそれぞれ次のようになります．




リロケーションコード(インタプリタの場合)

&H00 ……………… VARPTR で得たセグメントにデータがある.

&H00 以外……… SEGPTR(7) で得たセグメントにデータがある.

リロケーションコード(コンパイラの場合)

&H01 …………SEGPTR(8) で得たセグメントにデータがある.

&H01 以外 ……SEGPTR(7) で得たセグメントにデータがある.

# 第2部

## N88-日本語BASIC(86)
## コンパイラシステム

# 第1章

# コンパイラシステムの概要

## 1.1 コンパイラシステムの提供形態

$N_{88}$-日本語 BASIC(86) コンパイルシステムには，コンパイラおよびランタイムサポートパッケージの2つの媒体があります．

BASIC コンパイラのディスク(提供媒体)内のファイルとその内容を次の表に示します．

| ファイル名 | 内　容 |
|---|---|
| BASICC.COM | $N_{88}$-日本語 BASIC(86) コンパイラの本体 |
| SETUP.INI<br>ICP.EXE<br>INSTALL.CMF<br>INSTALL.BAT | BASIC を MS-DOS のシステムディスクに組み込む際に使用するファイル* |

*詳細については，第1部の「12.1　BASIC の登録」を参照してください．

ランタイムサポートパッケージのディスク(提供媒体)内のファイルとその内容を次の表に示します．なお，コンパイラによってコンパイルされたプログラムをオブジェクト(プログラム)と呼びます．

| ファイル名 | 内　容 |
|---|---|
| N88BASIC.LIB | オブジェクトプログラムの実行時に使用されるランタイムライブラリ．ただし，カレントドライブのカレントディレクトリあるいはルートディレクトリに格納されていなければならない |
| LIBMAINT.COM | ランタイムライブラリの機能強化の際に使用するユーティリティ．通常は使用しないため，そのまま保存しておく |
| N88SETUP.EXE | 画面ハードコピー機能，演算モード，RS-232C を設定するユーティリティ* |

*詳細については第1部の「12.2　セットアップユーティリティ」を参照してください．

## 1.2 コンパイラシステムの実現方式

### ■実現方式

一般にコンパイラシステムの実現方式には次の2つの方式があります．

#### ● Native-code 生成方式

コンパイラはオブジェクトコードとしてCPUが直接解釈実行可能な Native-code(機械語)を生成します．次にその特徴を挙げます．

- CPUが直接解釈実行するので，プログラムの高速な動作が可能．
- オブジェクトプログラムは，MS-DOS上で単体のプログラムとして実行できる．
- オブジェクトプログラムサイズが大きくなる．
- 割り込み処理を実現しにくい．

#### ● P-code 生成方式

コンパイラはオブジェクトコードとしてP-codeと呼ばれる中間コードを生成します．この中間コードはP-codeインタプリタによって解釈実行されます．

P-codeは，$N_{88}$-日本語BASIC(86)(MS-DOS版)のような一般のインタプリタが解釈実行する中間コードよりさらにコード化が進んだものです．

たとえば，参照される変数のアドレスは実行時のメモリレイアウトに対応するよう，コンパイル時にすでにアドレス解決されていますので，インタプリタのように変数が現れるたびにシンボルテーブルを探す必要がありません．次にその特徴を挙げます．

- インタプリタより効率化されているので，プログラムの高速な動作が可能．
- オブジェクトプログラムサイズが小さくなる．
- MS-DOS上でオブジェクトプログラムを実行する際，ランタイムライブラリが必要．
- 割り込み処理を実現しやすい．
- インタプリタと同じレベルの機能が実現可能．

$N_{88}$-日本語BASIC(86)コンパイラでは，P-code生成方式を採用しています．

### ■コンパイラが作成するオブジェクトプログラム

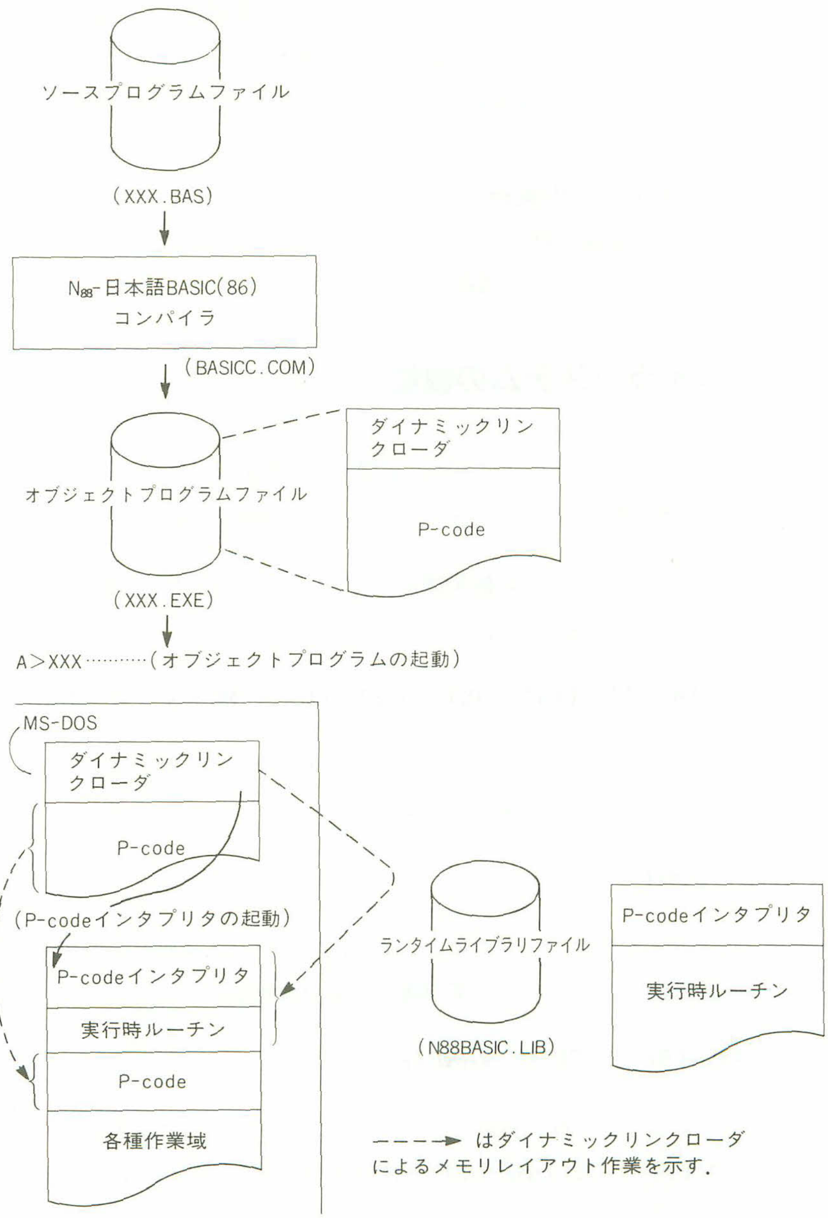
$N_{88}$-日本語BASIC(86)コンパイラが作成するオブジェクトプログラムはEXE形式ファイルとして出力されます．

したがって，コンパイラが出力したオブジェクトプログラムは，リンカを通さずに直接，MS-DOS上で実行させることができます．

■オブジェクトプログラムとランタイムライブラリ

オブジェクトプログラムを実行する際にはランタイムライブラリ(N88BASIC.LIB)が必要となります.

ランタイムライブラリにはオブジェクトプログラムを解釈実行する P-code インタプリタと入力ルーチン，演算ルーチン等の各種実行時ルーチンが格納されています.



### ■ダイナミックリンクローダ

オブジェクトプログラムにはP-codeの他にダイナミックリンクローダが格納されています．オブジェクトプログラム(EXE形式ファイル)が起動されると，最初にダイナミックリンクローダに制御が移ります．

ダイナミックリンクローダはランタイムライブラリファイルからP-codeインタプリタおよび実行時ルーチンを引き出し，P-code，P-codeインタプリタおよび各種実行時ルーチンをメモリに配置した後，P-codeインタプリタを起動します．

なお，ダイナミックリンクローダがメモリ上で占めていた領域は，P-codeインタプリタに制御がわたった後，オブジェクトプログラム実行時の作業用領域として使用されます．したがって，ダイナミックリンクローダが利用者メモリを圧迫するということはありません．

### ■オブジェクトプログラムの実行

P-codeインタプリタに制御が移ると，オブジェクトプログラムの実行が始まります．P-codeインタプリタはP-codeを一命令ずつ解釈実行します．

## 1.3　コンパイラシステムの機能

$N_{88}$-日本語BASIC(86)コンパイラシステムが提供する機能は，次に示す点を除き$N_{88}$-日本語BASIC(86)(MS-DOS版)インタプリタシステムが持つ機能と同等です．

・コンパイラシステムはプログラムの編集機能(スクリーンエディタ)を持っていません．したがって，次の命令は使用できません．

AUTO, DELETE, EDIT, LIST, LLIST, LOAD, MERGE, RENUM, SAVE

・コンパイラシステムにはインタプリタシステムのダイレクトモードに対応するモードがありません．したがって，次の命令は使用できません．

NEW, CONT

・コンパイラシステムとインタプリタシステムでは，プログラム実行時のメモリレイアウトが異なります．したがって，次のメモリ関連機能に多少の機能差があります．

CLEAR, FRE, SEGPTR, VARPTR

・コンパイラシステムでは各変数の型がプログラムのコンパイラ時に決定されます．したがって，DEFINT/DEFSNG/DEFDBL/DEFSTR命令等の型宣言命令は非実行命令になります．

・プログラムの連結機能(CHAIN および COMMON 命令)の一部が異なります．CHAIN 命令で MEARG オプションを使用することはできません．
・その他，CHILD，DRAW，PLAY，RUN，STOP 命令などに多少の機能差があります．

▶参照　インタプリタとコンパイラの機能差→「付録C　インタプリタとコンパイラの機能差一覧」
各命令の機能→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

なお，オブジェクトプログラム実行時の実行速度性能はおよそ次のようなものです．

・演算処理，文字列操作等が中心のプログラムの場合，実行速度はインタプリタシステムのおよそ 2～3 倍です．
・グラフィックを中心としたプログラムの場合，実行速度はインタプリタシステムのおよそ 2～3 倍です．
・ディスクに対する入出力を中心とした場合，実行速度はインタプリタシステムのおよそ 2 倍です．

# 第2章

# ソースプログラムの作成

## 2.1 ソースプログラム

$N_{88}$-日本語 BASIC(86)コンパイラに入力することのできるソースプログラムファイルは次のようなものです．

・$N_{88}$-日本語 BASIC(86)(MS-DOS 版)でセーブされたプログラムファイル

例） SAVE "TEST" ……………………バイナリセーブされたファイル
　　SAVE "TEST", A ………………アスキーセーブされたファイル
　　SAVE "TEST", P ………………プロテクトセーブされたファイル

・MS-DOS 上で動作する種々のエディタで作成した BASIC プログラムファイル
・ファイル変換ユーティリティで BASIC 形式から MS-DOS 形式に変換したプログラムファイル

$N_{88}$-日本語 BASIC(86)(MS-DOS 版)インタプリタは強力なスクリーンエディタ機能を持っています．ソースプログラムは，一般に $N_{88}$-日本語 BASIC(86)(MS-DOS 版)インタプリタシステムの環境で作成，修正を行って，コンパイラに移行するのがよいでしょう．

## 2.2 ファイル使用のための制御命令

インタプリタシステムにおいてはファイル同時オープン数およびランダムファイルのレコードサイズの宣言は起動パラメータ(/F：および/S：)で行いますが，コンパイラシステムにおいては，ソースプログラム中にその記述を行います．

REM △$FILE〔〈ファイル数〉〕〔，〈レコードサイズ〉〕
(△は空白を意味します)

〈ファイル数〉にはファイル同時オープン数を，〈レコードサイズ〉にはランダムファイルのレコードサイズを，それぞれ整数で指定します．指定できる値はインタプリタに準じます．

▶**参照**　ファイル同時オープン数，ランダムファイルのレコードサイズ→第1部の「2.3　起動パラメータの指定方法」

REM　$FILE命令は非実行命令であり，コンパイラによってのみ解釈されます．ただし1つのプログラム中に二度以上指定することはできません．

REM　$FILE命令を使用しない場合，〈ファイル数〉は2，〈レコードサイズ〉は256となります．

$FILEの後に注釈を記述する場合には，コロン(：)で区切ります．

例1)　REM　$FILE, 128

ファイル同時オープン数は2，レコードサイズは128バイトになります．

例2)　REM　$FILE　4, 64：FILE(4), RECORD(64 byte)

ファイル同時オープン数は4，レコードサイズは64バイトになります．

## 2.3　ソースプログラム作成上の注意点

ソースプログラムを作成する際にはコンパイラの処理能力にかかわる次のような制限を守らなければなりません．ただし，これらはコンパイラとしての処理能力からの制限であって，オブジェクトプログラム実行時の処理能力はメモリサイズなどの種々の環境により変化します．

・コンパイル可能なソースプログラムの行数はインタプリタとほぼ同等です．プログラムの記述方法によって変動がありますが，およそ2000行です．
・プログラム中で使用可能な変数名あるいはラベル等の名前の数はおよそ2500個(名前の長さを10文字平均とした場合の個数)です．
・FOR〜NEXT命令の個数の上限はおよそ300個です．
・WHILE〜WEND命令の個数の上限はおよそ300個です．
・1つの配列の大きさの上限は65530－(2×次元数)バイトです．

# 第3章

# ソースプログラムのコンパイル

## 3.1 コンパイラの起動方法

$N_{88}$-日本語 BASIC(86)コンパイラのプログラム名は〝BASICC.COM〞です．コンパイラの起動方法は次のとおりです．

A>BASICC 〔〈ソースプログラムファイル名〉〕〔,〈オブジェクトプログラムファイル名〉〕〔, リスティングファイル名〉〕〔/132〕〔/G〕

(A>は MS-DOS のプロンプト)

### ■ファイル名の入力

〈ソースプログラムファイル名〉,〈オブジェクトプログラムファイル名〉,〈リスティングファイル名〉は，いずれも次の形式で指定します．

〔〈パス名〉〕〈ファイル名〉

〈パス名〉はファイルの格納されているドライブとディレクトリを指定します．省略された場合，カレントドライブのカレントディレクトリに格納されているファイルが処理対象になります．

〈ファイル名〉は対象となるファイルのファイル名です．

また，〈ソースプログラムファイル名〉および〈オブジェクトプログラムファイル名〉の後にセミコロン(；)を指定すると，以降のパラメータには規定値が採用されます．

・〈ソースプログラムファイル名〉

コンパイルする BASIC のソースプログラムファイルです．

〈ファイル名〉には拡張子が指定できますが，拡張子の省略時には自動的に〝.BAS〞が指定されたとみなされます．

〈ソースプログラムファイル名〉を省略すると，コンパイラ起動後にソースプログラムの名前を要求する次のようなプロンプトメッセージが表示されますので，同様に指定します．この場合，〈ファイル名〉は必ず指定しなければなりません．

Source filename [.BAS]：〈ソースプログラムファイル名〉

・**〈オブジェクトプログラムファイル名〉**

コンパイル後のオブジェクトプログラムファイル(EXE形式)の名前です.

〈ファイル名〉には拡張子が指定できますが，拡張子の省略時には自動的に〝.EXE″が指定されたとみなされます.

〈オブジェクトプログラムファイル名〉を省略すると，コンパイラ起動後にオブジェクトプログラムの名前を要求する次のようなプロンプトメッセージが表示されますので，同様に指定します.

Object filename ［XXXX.EXE］:〈オブジェクトプログラムファイル名〉

このとき，〈オブジェクトプログラム名〉を省略すると，先に指定した〈ソースプログラムファイル名〉の拡張子を〝EXE″にしたファイル名が採用されます.

・**〈リスティングファイル名〉**

コンパイルリストを出力するファイルの名前です.

リスティングファイルには,一般のディスクファイル名のほか,CON(画面にコンパイルリストを表示する)，LST(プリンタにコンパイルリストを出力する)を指定することができます.

〈リスティングファイル名〉を省略すると，コンパイラ起動後にリスティングファイルの名前を要求する次のようなプロンプトメッセージが表示されますので，同様に指定します.

Source listing ［NUL.LST］:〈リスティングファイル名〉

このとき，〈リスティングファイル名〉を省略すると，コンパイルリストはどのファイル／装置にも出力されません.

なお，コンパイルリストには入力プログラムのソースリストと診断情報が出力されます.P-codeのリストは出力されません.

## ■コンパイラスイッチ

各ファイル名指定の後にはコンパイラスイッチを指定することができます.コンパイラスイッチの指定は，起動時のファイル名指定でも，それ以外のファイル名指定のときでもかまいません.コンパイラスイッチは次のとおりです.

| | |
|---|---|
| /132 | :リスティングファイルを132桁で出力します. |
| /G | :1バイト系グラフィック文字がソースプログラムに含まれている場合に指定します. |

日本語文字を含むソースプログラムに対し/Gを指定すると日本語が正しく解釈されなくなります.

## ■画面メッセージ

$N_{88}$-日本語BASIC(86)コンパイラが起動されると，次のメッセージが表示され，続いてコンパイルが始まります.

```
N88-BASIC(86) Compiler version x.x
Copyright (C) 1984,xx by NEC Corporation
```

コンパイルが正常に終了すると次のメッセージが表示されます.

```
*** NO ERROR ***

 <AREA SIZE>

   C O D E  :   nnnnn BYTES ─────────P-codeのサイズです
   VARIABLE :     nnn BYTES ─────────変数情報のサイズです
   CONSTANT :     nnn BYTES ─────────定数セグメントのサイズです

 XXXX SUCCESSFULLY COMPILED
```

次にいくつかの例を示します.

例1）パラメータを指定しない場合.

ソースプログラム"TEST.BAS"は，カレントドライブのカレントディレクトリに格納されています.

オブジェクトプログラム"TEST.EXE"をカレントドライブのカレントディレクトリに生成します.

コンパイルリストは出力されません.

```
A>BASICC ⏎

N88-BASIC(86) Compiler version x.x
Copyright (C) 1984,xx by NEC Corporation

Source filename [.BAS]: TEST ⏎ ─────────ソースプログラムファイル名を入力
Object filename [TEST.EXE]: ⏎ ─────────オブジェクトプログラム名はTEST.EXE
Source listing  [NUL.LST]: ⏎ ─────────コンパイルリストは出力しない

*** NO ERROR ***

 <AREA SIZE>

   C O D E  :   15248 BYTES
   VARIABLE :     335 BYTES
   CONSTANT :     662 BYTES

 TEST SUCCESSFULLY COMPILED

                                  (下線部分が入力した部分です)
```

**例2）** パラメータを設定し，コンパイルリストを画面に出力する場合．

カレントドライブのカレントディレクトリに格納されているソースプログラム"TEST.BAS"をコンパイルし，カレントドライブのカレントディレクトリにオブジェクトプログラム"TEST.EXE"を生成します．

コンパイルリストはコンソール(画面)に出力されます．この"TEST.BAS"には構文エラーがあるので，診断メッセージリストが出力されます．

```
A>BASICC ⏎

N88-BASIC(86) Compiler version x.x
Copyright (C) 1984,xx by NEC Corporation

Source filename [.BAS]: TEST ⏎ ――――――――ソースプログラムファイル名を入力
Object filename [TEST.EXE]: ⏎ ―――――――オブジェクトプログラム名はTEST.EXE
Source listing  [NUL.LST]: CON ⏎ ――――――コンパイルリストを画面に出力する


N88-BASIC(86) COMPILER  REV:x.x   PROGRAM-NAME:TEST      DATE:xx-xx-xx       ┐
                                                         TIME:15:09:18  PAGE:001 ├ コンパイラ
                                                                             │ リスト
....*....1....*....2....*....3....*....4....*....5....………               ┘
                                                                             ┐
10 TIME$="00:00:00"                                                          │
20 FORI=1 TO 10000:I=I+1:J=I^.5:NEXT                                         │
30 PRINT TIME$                                                               │
                                                                             │
                                                                             │
N88-BASIC(86) COMPILER  REV:x.x   PROGRAM-NAME:TEST      DATE:xx-xx-xx       │
                                                         TIME:15:09:18  PAGE:002 │
                                                                             │
 LINE  L  DIAGNO *** DIAGNOSTIC MESSAGE ***                                  ├ エラー
                                                                             │ メッセージ
   20  F     2   SYNTAX ERROR                                                │
   20  F     1   NEXT WITHOUT FOR                                            │
                                                                             │
 <ERROR SUMMARY>                                                             │
                                                                             │
    0002  FATAL ERRORS                                                       │
    0000  WARNINGS                                                           │
                                                                             │
 TEST UNSUCCESSFULLY COMPILED                                                ┘
```

（下線部分が入力した部分です）

**例3）** 起動時にソースプログラムファイル名を指定する場合.

起動時にソースプログラムファイル名〝TEST.BAS〟を指定します．このファイルはカレントドライブのカレントディレクトリに格納されているファイルです．

オブジェクトファイルは〝TEST.EXE〟になります．このファイルはカレントドライブのカレントディレクトリに格納されます．

コンパイルリストはコンソールに出力されません．

```
A>BASICC TEST; ⏎

N88-BASIC(86) Compiler version x.x
Copyright (C) 1984,xx by NEC Corporation

*** NO ERROR ***

 <AREA SIZE>

   C O D E  :      80 BYTES
   VARIABLE :      10 BYTES
   CONSTANT :      12 BYTES

 TEST SUCCESSFULLY COMPILED
```

（下線部分が入力した部分です）

**例4）** コンパイラスイッチを指定した場合.

ソースプログラムファイル名は〝TEST.BAS〟です．このファイルはカレントドライブのカレントディレクトリに格納されているファイルです．

このとき，コンパイラスイッチ/Gを指定していますので，グラフィックス文字は正しく解釈されますが，日本語が正しく解釈されません．また，コンパイラスイッチ/132を指定していますので，リスティングファイルは132桁で出力されます．

オブジェクトファイルはコンソールから入力します．この場合，〝TEST.EXE〟です．このファイルはカレントドライブのカレントディレクトリに格納されます．

コンパイルリストは出力されません．

```
A>BASICC ⏎

N88-BASIC(86) Compiler version x.x
Copyright (C) 1984,xx by NEC Corporation

Source filename [.BAS]: TEST/G/132 ⏎
Object filename [TEST.EXE]: ⏎
Source listing  [NUL.LST]: ⏎

*** NO ERROR ***

 <AREA SIZE>

   C O D E  :      80 BYTES
   VARIABLE :      10 BYTES
   CONSTANT :      12 BYTES

 TEST SUCCESSFULLY COMPILED
```

（下線部分が入力した部分です）

# 3.2 エラーメッセージ

## ■コンパイラ起動時のエラー

コンパイラが起動できない状態です.

次にエラーの一覧を示します.

| エラーメッセージ | エラーの内容 |
| --- | --- |
| メモリーが足りない | コンパイラの起動に必要なメモリ容量が足りない |
| パラメータ　エラー | パラメータの指定に誤りがある |
| コンパイル　スイッチ　エラー | コンパイルスイッチの指定に誤りがある |

## ■コンパイル時のエラー

### ●重大(Serious)エラー

このエラーが起こるとコンパイルが中止され，オブジェクトプログラムファイルは生成されません.

次にエラーの一覧を示します.

| エラーメッセージ | エラーの内容 |
| --- | --- |
| ソースファイルが見つからない | ソースファイルの指定に誤りがある |
| ソースファイルが空である | 指定されたソースファイルの内容が空である |
| ソースファイルが正しくない | BASICインタプリタあるいはテキストエディタが出力したファイルではない |
| ソースファイルがオープンできない | 指定されたソースファイルのパス指定に誤りがありオープンできない |
| ソースファイルリードエラー | ソースファイルの読み込みが正常に行えない |
| ソースファイルが大きすぎる | ソースファイルが大きすぎてコンパイルできない |
| オブジェクトファイルが作成できない. | オブジェクトファイルがオープンできないか，あるいはオブジェクトファイルの出力先がディスクファイルではない |
| オブジェクトファイルライトエラー | オブジェクトファイルへの書き出しが正常に行えない |
| リストファイルが作成できない | リスティングファイルのパス指定に誤りがありオープンできない |
| リストファイルファイルライトエラー | リスティングファイルへの書き出しが正常に行えない |
| FOR-NEXTが多すぎる | FOR～NEXT命令の数が多すぎる |
| WHILE-WENDが多すぎる | WHILE～WEND命令の数が多すぎる |
| メモリーが足りない | シンボル，文字列定数，DATA命令などが多すぎるか，あるいは命令，式が複雑すぎる |

●文法エラー

このエラーが起こってもコンパイルは続行され，最後にまとめてエラーメッセージが出力されます．文法エラーには，致命的エラーと警告エラーの2種類があります．

エラーの種類，エラー番号，エラーメッセージ(英文)はインタプリタと同様です．

▶参照　文法エラー→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

●致命的(Fatal)エラー

このエラーがあるとオブジェクトファイルは作成されません．

次にコンパイラ固有のエラーの内容を次に示します．

| エラー番号 | エラーメッセージ | エラーの内容 |
|---|---|---|
| 200 | ELSE WITHOUT IF | ELSE が IF 以外のところに使用されている |
| 201 | DUPLICATE LINE NUMBER | 行番号が2重定義されている |
| 202 | ILLEGAL LINE NUMBER | 行番号の形式が不正である |
| 203 | PROGRAM TOO LARGE | ソースプログラムの行数が多すぎる |
| 204 | $FILE STATEMENT ERROR | REM　$FILE 命令の指定が不正である |
| 205 | STATEMENT TOO COMPLEX | 式の構造が複雑すぎる |

**●警告(Warning)エラー**

このエラーが起こってもオブジェクトファイルは作成されます．エラーが発生した命令はそのまま P-code に変換されますが，オブジェクトプログラム実行時にも同様のエラーが発生します．

次に警告エラーの内容を示します．

| エラー番号 | エラーメッセージ | エラーの内容 |
|---|---|---|
| 5 | ILLEGAL FUNCTION CALL | 命令の使い方に誤りがある |
| 11 | DIVISION BY 0 | ゼロで除算を行った |
| 50 | FIELD OVERFLOW | フィールドの長さがレコードサイズを超えている |
| 230 | SAME VARIABLE IN DEF FN PARAMETER | DEF FN 命令のパラメータに同一変数名が存在している |
| 231 | SAME VARIABLE IN COMMON STATEMENT | COMMON 命令のパラメータに同一変数名が存在している |
| 232 | FILE NUMBER TOO LARGE | 同時オープン可能数を超えたファイル番号が使用されている |
| 233 | FOR-NEXT IDENTIFIER MISMATCH | 対応する FOR〜NEXT 命令の制御変数名が一致していない |
| 234 | INADEQUATE POSITION OF DEFXXX | DEFINT/SNG/DBL/STR 命令で指定された英字はすでに型定義されている |

# 第4章

# オブジェクトプログラムの実行

## 4.1　オブジェクトプログラムの実行方法

### ■ランタイムライブラリファイルの準備

オブジェクトプログラムの実行を行う場合，ランタイムライブラリ(N88BASIC.LIB)が必要です．

ランタイムライブラリはカレントドライブのカレントディレクトリあるいはルートディレクトリのいずれかに登録されていなければなりません．

ランタイムライブラリが用意されていない場合，エラーとなり，オブジェクトプログラムは実行できません．

ランタイムライブラリはランタイムサポートパッケージの提供媒体内に格納されています．

### ■オブジェクトプログラムの実行

コンパイラが出力するオブジェクトプログラムファイルはEXE形式のファイルですから，MS-DOS上でそのまま実行可能です．

```
A>BASICC ⏎

N88-BASIC(86) Compiler version x.x
Copyright (C) 1984,xx by NEC Corporation

Source filename [.BAS]: HAKO ⏎
Object filename [HAKO.EXE]: ⏎
Source listing  [NUL.LST]: ⏎

*** NO ERROR ***

 <AREA SIZE>

    C O D E  :    15248 BYTES
    VARIABLE :      335 BYTES
    CONSTANT :      662 BYTES

 HAKO SUCCESSFULLY COMPILED

A>HAKO ⏎ ――――――――――――オブジェクトプログラムの実行
```

（下線部分が入力した部分です）

## ■オブジェクトプログラム実行開始直後の画面

オブジェクトプログラムの実行が始まると，まず画面が消去されます．このときスクロール範囲は画面全体となっています．

ファンクションキーは表示されません．必要な場合はCONSOLE命令で表示させるようにしてください．

以降，画面の制御はオブジェクトプログラムに移行します．

## ■ オブジェクトプログラム実行終了時の画面

プログラムの終了方法により画面の状態が異なります．

**・END 命令，SYSTEM 命令の実行時，あるいはプログラムの終端に達した場合**

画面全体が消去され，MS-DOS のプロンプトが表示されます．

**・STOP 命令の実行時，STOP キーの押下時，あるいはエラーの発生時**

画面の状態はそのままで次のメッセージが表示され，キー入力待ちとなります．

```
Press any key to command mode
```

任意のキーを押すと，画面は消去され，MS-DOS のプロンプトが表示されます．

# 4.2 エラーメッセージ

## ■オブジェクトプログラム実行開始前に発生するエラー

これらのエラーが発生するとオブジェクトプログラムは実行されません.

| エラーメッセージ | エラーの内容と対策 |
|---|---|
| Out of memory | オブジェクトプログラムのサイズが大きすぎてメモリに格納できない. メモリ容量を拡大するか, プログラムを縮小する |
| Too meny files | REM $FILE 命令で指定された同時ファイルオープン数をシステムで扱うことができない<br>"CONFIG . SYS"ファイル FILES パラメータの値を大きくする<br>ただし, REM $FILE命令で指定できる値は0～13 |
| File not found "N88BASIC . LIB" | ランタイムライブラリ "N88BASIC . LIB" がカレントドライブのカレントディレクトリあるいはルートディレクトリに登録されていない |
| Invalid file "N88BASIC . LIB" | ランタイムライブラリ "N88BASIC . LIB" の内容が不正 |

## ■オブジェクトプログラム実行時のエラー

オブジェクトプログラム実行時に発生するエラーは基本的にインタプリタシステムでプログラムを実行する場合のエラーと同じです.

▶参照　エラー→「$N_{88}$-日本語 BASIC(86)(MS-DOS 版)リファレンスマニュアル」

次のエラーはコンパイルしたオブジェクトプログラムの実行時に発生する, コンパイラシステム固有のものです. インタプリタシステムでは発生しません.

| エラーメッセージ | エラーの内容 |
|---|---|
| Invalid file | CHAIN されるプログラムが BASIC コンパイラの出力したオブジェクトプログラムではない. |
| Unmatch common data | 呼び出される側の COMMON 命令の変数の並びが一致していない. |
| RUN can't execute | RUN 命令で他のプログラムを起動する際にコマンドプロセッサ(COMMAND . COM)が見つからない. |

# 付録

# 付録A

## N88-日本語BASIC(86)と
## N88-日本語BASIC(86)(MS-DOS版)との機能差一覧

N88-日本語BASIC(86)と異なり，N88-日本語BASIC(86)(MS-DOS版)インタプリタのノーマルモードはMS-DOS上で動作するため，機能の差があります．

ここでは，その機能上の差を次の一覧表にまとめています．

| 項番 | 機　　能 | N88-日本語BASIC(86)<br>(Ver 6.0) | N88-日本語BASIC(86)<br>(MS-DOS版)6.0 |
|---|---|---|---|
| 1 | OS | N88-日本語BASIC(86) | MS-DOS |
| 2 | ターミナルモード | あり | なし |
| 3 | モニタモード | あり | なし |
| 4 | ライトペン関連機能 | あり | なし |
| 5 | マウス関連機能 | マウスドライバによる制御 | マウス関連の命令・関数あり |
| 6 | チャイルドプロセス制御 | なし | あり |
| 7 | ウォームリスタート機能 | あり | なし |
| 8 | オートスタート機能 | システムディスクのID部にオートスタート情報を設定しておく | MS-DOSのAUTOEXEC. BATファイルを使用する |
| 9 | ディスクファイルの形式 | BASIC形式 | MS-DOS形式 |
| 10 | ディスク管理体系 | ファイル単位の管理 | MS-DOSの階層ディレクトリ構造による管理 |
| 11 | ディスクファイル名の指定形式 | “ドライブ番号：ファイル名”<br>例 “2：SUB1” | “ドライブ名：ディレクトリ名　ファイル名”<br>例 “B：SUB1”<br>　“B：¥BASIC¥SUB1” |
| 12 | ランダムファイルのレコードサイズ | 256バイト固定 | 1～2048バイトの範囲で指定できる<br>起動パラメータ/S：で指定する |
| 13 | ファイル同時オープン数 | 最大15個<br>“How many files?”メッセージに対して応答する | 最大13個<br>起動パラメータ/F：で指定する |
| 14 | ディスクファイル内の日本語コード | JISコード<br>(KI/KOコード挿入方式) | シフトJISコード |
| 15 | ファイルの共用制御 | なし | MS-DOS（Ver 3.1以降)＋MS-NETWORKSの環境で使用可能 |
| 16 | 日本語入力方法 | AI逐次，AI連文節，逐次，連文節，単文節変換入力，JISコード入力<br>辞書ファイル：BASIC独自<br>操作方法：BASIC独自 | MS-DOSに依存<br>辞書ファイル：MS-DOSと共用<br>操作方法：MS-DOSの日本語入力方法に準じる |

| 項番 | 機　　能 | N88-日本語 BASIC(86)<br>(Ver 6.0) | N88-日本語 BASIC(86)<br>(MS-DOS 版)6.0 |
|---|---|---|---|
| 17 | 利用者定義文字の扱い | mkfont. n88 ユーティリティにより usfont. n88 ファイルに対して登録する | USKCGM コマンドにより USKCG. SYS ファイルに対して登録する(MS-DOS と共用) |
| 18 | 日本語文字と1バイト系グラフィック文字の混在使用 | 混在使用可能 | 混在使用不可能<br>日本語文字を扱う日本語キャラクターモードとグラフィック文字を扱うグラフィックキャラクタモードがある |
| 19 | メモリレイアウト | メモリ上限<br>機械語プログラム領域<br>配列データ領域<br>文字列演算用作業域<br>文字列データ領域<br>シンボルテーブル<br>DISK CODE<br>利用者スタック領域<br>システムスタック領域<br>プログラム領域<br>システム制御情報／入出力バッファ<br>システム制御情報<br>0番地 | メモリ上限<br>チャイルドプロセス領域<br>機械語プログラム領域<br>配列データ領域<br>文字列演算用作業域<br>文字列データ領域<br>シンボルテーブル<br>利用者スタック領域<br>システムスタック領域<br>プログラム領域<br>システム制御情報／入出力バッファ<br>インタプリタシステムコード (2)<br>インタプリタシステムコード (1)<br>MS-DOS<br>0番地 |
| 20 | 機械語プログラムの形式 | 先頭に4バイトの制御コードが付加されている | 制御コードは付加されない |
| 21 | 拡張機能の使用方法 | switch. n88 ユーティリティによりメモリスイッチに対して登録する<br>10進演算制御機能，BRANCH 4670 対応ネットワーク制御機能，MS-NETWORKS 制御機能は使用できない | 起動パラメータ/E：で使用の有無を宣言する |
| 22 | 画面ハードコピー | switch. n88 ユーティリティによりメモリスイッチでハードコピーの種類を設定し，さらに setup. n88 ユーティリティにより機能を選択する | セットアップユーティリティ N88SETUP. EXE によりハードコピーの種類と機能を選択する（本体のメモリスイッチの値によらない） |
| 23 | BLOAD 命令 | 〈ロードアドレス〉指定が省略された場合，BSAVE 命令実行時に指定された相対アドレスの位置に機械語プログラムがロードされる | 〈ロードアドレス〉が省略された場合，機械語プログラム領域の先頭から機械語プログラムがロードされる |
| 24 | CHILD 命令 | なし | チャイルドプロセスを実行する |

| 項番 | 機　　能 | $N_{88}$-日本語 BASIC(86)<br>(Ver 6.0) | $N_{88}$-日本語 BASIC(86)<br>(MS-DOS 版)6.0 |
|---|---|---|---|
| 25 | CLEAR 命令 | CLEAR〔〈ストリング領域の大きさ〉〕〔,〈メモリの上限〉〕〔,〈スタックの大きさ〉〕〔,〈配列データ領域の大きさ〉〕 | CLEAR〔〈機械語プログラム領域のサイズ〉〕〔,〈利用者スタック領域のサイズ〉〕〔,〈配列データ領域のサイズ〉〕〔,〈チャイルドプロセス領域のサイズ〉〕 |
| 26 | CONSOLE 命令 | 〈キャラクタモードスイッチ〉の指定なし | 〈キャラクタモードスイッチ〉で日本語キャラクタモードとグラフィックキャラクタモードの指定を行う |
| 27 | DSKF 関数 | 対象ディスクの各種諸元を得る | 対象ディスクの未使用領域のサイズを得る |
| 28 | DSKI$，DSKO$ 関数 | あり | なし |
| 29 | FILED 命令 | バッファサイズは 256 バイト固定 | バッファサイズは 1～2048 バイトの範囲で指定できる<br>起動パラメータ/S：で指定する |
| 30 | FILES 命令 | ファイルサイズの単位：クラスタ<br>ファイル作成日時の出力：なし | ファイルサイズの単位：バイト<br>ファイル作成日時の出力：あり |
| 31 | GET/PUT 命令 | ランダムファイルのレコード番号：<br>1～65000 | ランダムファイルのレコード番号：<br>1～65535 |
| 32 | KINPUT 命令 | プロンプトとして〝?〟が無条件に表示される | プロンプトは表示されない |
| 33 | KLEN 関数 | 〈機能〉の値は 0，1，2，3，4，5 | 〈機能〉の関数値は 0，1，2，3，4 |
| 34 | KTYPE 関数 | 得られる値は 0，1，2，3，4 | 得られる値は 0，1，2 |
| 35 | LOF 関数 | ファイルの大きさの単位はセクタ | ファイルの大きさの単位はバイト |
| 36 | MKDIR，CHDIR，RMDIR 命令 | なし | ディスクのディレクトリを管理する |
| 37 | NEW ON 命令 | あり | なし |
| 38 | OPEN 命令 | 〈共用モード〉指定なし | 〈共用モード〉指定あり |
| 39 | SEGPTR 関数 | なし | 各種制御情報領域のセグメントベースアドレスを得る |
| 40 | SET 命令/ATTR$ 関数 | R(Read after write)属性と<br>P(Protect)属性 | P(Protect)属性のみ |
| 41 | STATUS SYSTEM 関数 | なし | ハードウェアの諸元を得る |
| 42 | SYSTEM 命令 | なし | MS-DOS に制御をもどす |

# 付録B

## ノーマルモードとハイレゾモードとの機能差一覧

ノーマルモードとハイレゾモードでは，ハードウェア上の制約で，機能の差があります．ここでは，その機能上の差を次の一覧表にまとめています．

| 項番 | 機　能 | ハイレゾモード | ノーマルモード |
|---|---|---|---|
| 1 | グラフィック画面サイズ<br>(オリジナルスクリーン座標系) | 1120×936 | 640×400<br>640×200 |
| 2 | 表示画面サイズ | 1120×750 ドット | 640×400 ドット<br>640×200 ドット |
| 3 | テキスト画面サイズ | 80 桁×25 行 | {40, 80} 桁×{20, 25} 行 |
| 4 | 表示モード | 1120×936　カラー　1ページ<br>1120×936　白黒　4ページ | 640×400　カラー　2ページ<br>640×400　白黒　6(8)ページ<br>640×200　カラー　4ページ<br>640×200　白黒　12(16)ページ<br>(　)内は 4096 色中･16 色モードのとき |
| 5 | カラー | 4096 色中・16 色の表示が可能 | 8 色中･8 色，4096 色中･8 色，4096 色中･16 色のいずれかの表示が可能 |
| 6 | パレット | 0～15 | 0～7<br>0～15 (4096 色中･16 色モードのとき) |
| 7 | タイルストリングの文字列長 | カラーモード時：<br>4×n 文字 | カラーモード時：<br>3×n 文字<br>4×n 文字(4096 色中・16 色モードのときのみ) |
| 8 | 白黒モード時の濃淡指定 | 各ドット毎に 16 色階調中 2 種類の濃淡を表示することが可能 | なし |
| 9 | 〔2〕COLOR 命令 | パラメータを全部省略した場合，パレットは初期状態にもどる | 基本グラフィックモードでパラメータを省略した場合，エラーとなる |
| 10 | COPY 命令 | カラープリンタ対応：なし | カラープリンタ対応：あり |
| 11 | KPLOAD 命令 | 文字パターンサイズ：<br>24×24 ドット | 文字パターンサイズ：<br>16×16 ドット |

| 項番 | 機　　能 | ハイレゾモード | ノーマルモード |
|---|---|---|---|
| 12 | MOUSE 命令 | マウスカーソルのサイズ：<br>32×32 ドット<br>マウスカーソルの色指定：<br>4096 色中 4 色を表示することが可能<br>マウスカーソルの形状：<br>XOR 文字列と AND 文字列で指定可能 | マウスカーソルのサイズ：<br>16×32 ドット<br>マウスカーソルの色指定：<br>8 色中・8 色モードおよび 4096 色中・8 色モードのとき 8 色中 3 色を表示することが可能<br>4096 色中・16 色モードのとき 4096 色中 4 色を表示することが可能<br>マウスカーソルの形状：<br>XOR 文字列で指定 |
| 13 | SCROOL 命令 | グラフィック画面(VRAM)の表示領域を設定する | なし |

# 付録C

## インタプリタとコンパイラの機能差一覧

インタプリタシステムとコンパイラシステムにおける各命令の機能差は「$N_{88}$-日本語 BASIC (86)(MS-DOS版)リファレンスマニュアルに命令単位で詳細に解説されています。

ここでは，その概要を次の表にまとめています。

メモリレイアウトについては，第1部の「第14章　メモリマップとメモリ管理」を参照してください。

| 項番 | 機　　能 | インタプリタ | コンパイラ<br>（またはオブジェクトプログラム） |
|---|---|---|---|
| 1 | ランダムファイルの<br>レコードサイズ | インタプリタ起動時の/S：パラメータで指定する | REM　$FILE 命令で指定する |
| 2 | 拡張機能 | インタフェースボードをセットし，起動時の/E：パラメータで使用宣言する | インタフェースボードをセットするだけでよい |
| 3 | 電話制御機能 | あり | 使用できない |
| 4 | 数値配列の場合の要素数 | 整数型　：32767 個<br>単精度型：16383 個<br>倍精度型：　8191 個 | （小数点以下は切上げ）<br>整数型　：32767−(次元数+3)個<br>単精度型：16383−(次元数+3)/2個<br>倍精度型：　8191−(次元数+3)/4個 |
| 5 | 画面ハードコピー | N88 SETUP. EXE ユーティリティで N88 BASIC. EXE に対して設定を行う | N88 SETUP. EXE ユーティリティで N88 BASIC. LIB に対して設定を行う |
| 6 | エラーの発生 | 実行時に発生する | コンパイル時に発生する場合と実行時に発生する場合がある |
| 7 | AUTO, CONT, DELETE, EDIT, LIST, LLIST, LOAD, MERGE, NEW, RENUM, SAVE 命令 | あり | 使用できない |
| 8 | CHAIN 命令 | (1) MERGE, 〈行番号〉, ALL, DELETE オプション指定可<br>(2) COMMON 命令は呼び出す側にのみ存在すればよい<br>(3) 呼び出されるプログラムはアスキー／バイナリ形式でセーブされた BASIC プログラム<br>(4) 呼び出されるプログラムのファイル名に拡張子の指定がない場合，BAS が付加される | (1) MERGE, 〈行番号〉, ALL, DELETE オプション指定不可<br>(2) COMMON 命令は呼び出す側／呼び出される側の両方に同一形式で必要<br>(3) 呼び出されるプログラムはコンパイルされたものでなければならない<br>(4) 呼び出されるプログラムのファイル名に拡張子の指定がない場合，EXE が付加される |

| 項番 | 機　　能 | インタプリタ | コンパイラ<br>（またはオブジェクトプログラム） |
|---|---|---|---|
| 9 | CHILD 命令 | M 指定時，チャイルドプロセスが終了するとダイレクトモードへもどる | M 指定時，チャイルドプロセスが終了すると MS-DOS へもどる |
| 10 | CLEAR 命令の第 2 パラメータ | 〈利用者スタック領域のサイズ〉 | 〈文字列領域／スタック領域のサイズ〉 |
| 11 | COMMON 命令 | CHAIN 命令実行以前に COMMON 命令が必要 | CHAIN 命令実行以前に COMMON 命令が必要<br>呼び出される側にも同一形式の COMMON 命令が必要 |
| 12 | CONSOLE 命令 | インタプリタ起動直後，ファンクションキーは表示されるモードになっている | プログラム実行直後，ファンクションキーは表示されないモードになっている |
| 13 | DEFINT/DEFSNG/<br>DEFDBL/DEFSTR 命令 | これらの命令は実行文<br>変数の型は動的に変更できる | これらの命令は非実行文<br>変数の型は DEFINT/SNG/DBL/STR の物理的位置関係できまる<br>(動的に変数の型は変更できない) |
| 14 | DRAW 命令 | =〈変更名〉；指定可<br>X 指定可 | =〈変更名〉；指定不可<br>X 指定不可 |
| 15 | END 命令 | プログラムの実行を終了し，ダイレクトモードへもどる | プログラムの実行を終了し，MS-DOS へもどる |
| 16 | FRE 関数 | 引数：機能<br>0：シンボルテーブル／文字列データ領域の未使用領域のサイズを得る<br>1：プログラム領域の未使用領域のサイズを得る<br>2：機能 0，1 の和を得る<br>3：配列データ領域の未使用領域サイズを得る<br>4：エラー<br>5：エラー<br>6：エラー | 引数：機能<br>0：文字列データ領域／スタック領域の未使用領域のサイズを得る<br>1：エラー<br>2：エラー<br>3：インタプリタと同様<br>4：文字列データ領域／スタック領域の全体サイズを得る<br>5：配列データ領域の全体サイズを得る<br>6：シンボルテーブル／システム制御情報／入出力バッファ領域の全体サイズを得る |
| 17 | OPEN 命令 | ファイル同時オープン数はインタプリタ起動時に/F：パラメータで指定する | ファイル同時オープン数は REM $FILE 命令で指定する |
| 18 | PLAY 命令 | =〈変数名〉；指定可 | =〈変数名〉；指定不可 |
| 19 | REM 命令 | 注釈の指定 | REM $FILE〔〈ファイル同時オープン数〉〕〔，〈レコードサイズ〉〕の指定の形式でランダムファイルレコードサイズ，ファイル同時オープン数を指定できる<br>それ以外の形式の場合は注釈 |

<table>
<tr><th>項番</th><th>機　　能</th><th>インタプリタ</th><th>コンパイラ<br>（またはオブジェクトプログラム）</th></tr>
<tr><td>20</td><td>RUN 命令</td><td>(1)　R 指定可<br><br>(2)　パス名の指定が可能<br><br>(3)　起動するプログラムは BASIC プログラムのみ<br><br>(4)　起動するプログラムのファイル名に拡張子の指定がない場合，BAS が採用される<br><br>(5)　呼び出す側のプログラムと呼び出されたプログラムがメモリ上で置き変わる</td><td>(1)　R 指定不可（ファイルはいったんクローズされた後，プログラムが起動される）<br>(2)　パス名の指定ができない（ENVIRON 命令の〝PATH〞指定による）<br>(3)　EXE 形式／COM 形式のプログラムまたは BAT 形式のバッチプログラムであれば起動するプログラムはどんな種類のプログラムでもかまわない<br>(4)　起動するプログラムのファイル名に拡張子の指定がない場合，COM，EXE，BAT の順でファイル名を検索し，最初に一致したファイル名が採用される<br>(5)　呼び出す側のプログラムの制御情報の一部メモリに残される</td></tr>
<tr><td>21</td><td>SEGPTR 関数</td><td>
<table>
<tr><th>引数</th><th>機　　能</th></tr>
<tr><td>0</td><td>メモリ上限のセグメントアドレスを得る</td></tr>
<tr><td>1</td><td>チャイルドプロセス領域のセグメントアドレスを得る</td></tr>
<tr><td>2</td><td>機械語領域のセグメントアドレスを得る</td></tr>
<tr><td>3</td><td>配列データ領域のセグメントアドレスを得る</td></tr>
<tr><td>4</td><td>シンボルテーブル／文字列領域の終端のセグメントアドレスを得る</td></tr>
<tr><td>5</td><td>シンボルテーブル／文字列領域の先頭のセグメントアドレスを得る</td></tr>
<tr><td>6</td><td>プログラム領域のセグメントアドレスを得る</td></tr>
<tr><td>7</td><td>システム制御情報領域のセグメントアドレスを得る</td></tr>
<tr><td>8</td><td>エラー</td></tr>
</table>
</td><td>
<table>
<tr><th>引数</th><th>機　　能</th></tr>
<tr><td>0</td><td>インタプリタと同様</td></tr>
<tr><td>1</td><td>インタプリタと同様</td></tr>
<tr><td>2</td><td>インタプリタと同様</td></tr>
<tr><td>3</td><td>インタプリタと同様</td></tr>
<tr><td>4</td><td>エラー</td></tr>
<tr><td>5</td><td>エラー</td></tr>
<tr><td>6</td><td>エラー</td></tr>
<tr><td>7</td><td>インタプリタと同様</td></tr>
<tr><td>8</td><td>定数領域のセグメントアドレスを得る</td></tr>
</table>
</td></tr>
<tr><td>22</td><td>STOP 命令</td><td>プログラムの処理を中断し，ダイレクトモードにもどる（CONT 命令実行可能）</td><td>プログラムの処理を中止し，次のメッセージを表示して，キー入力を待つ<br>〝Press any key to command mode〞<br>応答したら，MS-DOS へもどる（CONT 命令実行不可）</td></tr>
</table>

| 項番 | 機　　能 | インタプリタ | コンパイラ（またはオブジェクトプログラム） |
|---|---|---|---|
| 23 | VARPTR関数 | ストリングディスクリプタ内のリロケーションコードの値と意味<br>&H00：VARPTRで得たセグメントにデータが存在する<br>&H00以外：SEGPTR(7)で得たセグメントにデータが存在する | ストリングディスクリプタ内のリロケーションコードの値と意味<br>&H01：SEGPTR(8)で得たセグメントにデータが存在する<br>&H01以外：SEGPTR(7)で得たセグメントにデータが存在する |

# 付録D

# 数値演算プロセッサの使用

数値演算プロセッサは浮動小数点データによる演算を高速化するための付加プロセッサです.

数値演算プロセッサが本体に実装されている場合,MS-DOS の SWITCH コマンドで数値演算プロセッサの使用を設定(所定のメモリスイッチを設定)することによって,BASIC の数値演算の実行速度を速めることができます.

数値演算プロセッサを使用することによって高速化されるのは,四則演算を除く次の数値関数あるいは演算処理です.

SIN, COS, TAN, ATN, EXP, SQR, べき乗演算

---

**注意**:数値演算プロセッサの使用できない機種があります.

---

# 付録E

## キャラクタコード表

上位4ビット→

| 下位4ビット↓ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 |  | DE | SP | 0 | @ | P |  | p | ▁ | ┴ | SP | ー | タ | ミ | ═ | ╳ |
| 1 | SH | D1 | ! | 1 | A | Q | a | q | ▂ | ┬ | 。 | ア | チ | ム | ╞ | 円 |
| 2 | SX | D2 | ” | 2 | B | R | b | r | ▃ | ┤ | 「 | イ | ツ | メ | ╪ | 年 |
| 3 | EX | D3 | # | 3 | C | S | c | s | ▄ | ├ | 」 | ウ | テ | モ | ╡ | 月 |
| 4 | ET | D4 | $ | 4 | D | T | d | t | ▅ | ▔ | 、 | エ | ト | ヤ | ◢ | 日 |
| 5 | EQ | NK | % | 5 | E | U | e | u | ▆ | ─ | ・ | オ | ナ | ユ | ◤ | 時 |
| 6 | AK | SN | & | 6 | F | V | f | v | ▇ | │ | ヲ | カ | ニ | ヨ | ◥ | 分 |
| 7 | BL | EB | ’ | 7 | G | W | g | w | █ | ▕ | ァ | キ | ヌ | ラ | ◤ | 秒 |
| 8 | BS | CN | ( | 8 | H | X | h | x | ▉ | ┌ | ィ | ク | ネ | リ | ♠ |  |
| 9 | HT | EM | ) | 9 | I | Y | i | y | ▊ | ┐ | ゥ | ケ | ノ | ル | ♥ |  |
| A | LF | SB | * | : | J | Z | j | z | ▋ | └ | ェ | コ | ハ | レ | ♦ |  |
| B | HM | EC | + | ; | K | [ | k | { | ▌ | ┘ | ォ | サ | ヒ | ロ | ♣ |  |
| C | CL | → | , | < | L | ¥ | \| | ¦ | ▍ | ╭ | ャ | シ | フ | ワ | ● |  |
| D | CR | ← | − | = | M | ] | m | } | ▎ | ╮ | ュ | ス | ヘ | ン | ○ |  |
| E | SO | ↑ | . | > | N | ^ | n | ~ | ▏ | ╰ | ョ | セ | ホ | ゛ | ╱ |  |
| F | SI | ↓ | / | ? | O | _ | o |  | ┼ | ╯ | ッ | ソ | マ | ゜ | ╲ |  |

# 索 引

## 数字・記号



## A



## B



## C

し



す



せ



そ



た



ち



つ



て

**わ**

NEC